BIT-BASIC Luiz C. Bittencourt

Luiz Carlos Bittencourt

MSX

BIT-BASIC

uma ponte para o assembler

R.L
1/8/89

Luiz Carlos Bittencourt

(M S X)

B I T - B A S I C

UMA PONTE PARA O ASSEMBLER

INFORMÁTICA

CAIXA POSTAL 8127
CURITIBA - PR
CEP - 80000

Nº 0070

AUTOR : LUIZ CARLOS BITTENCOURT

EDITORAÇÃO : JUZELE BITTENCOURT

FICHA CATALOGRÁFICA

Bittencourt, Luiz Carlos, 1948-
(MSX) BIT-BASIC, uma ponte para o ASSEMBLER. Curitiba, 1988.
166 p.

1. BIT-BASIC (Linguagem programada para computadores). 2. ASSEMBLER (Linguagem programada para computadores). 3. Programação (Computadores eletrônicos). 4. Microcomputadores - Programação. I. Título.

CDD (19º ed.)
001.6425

SUMÁRIO

PRÓLOGO

Compõem esta publicação um SOFTWARE e um LIVRO.

BIT-BASIC

O SOFTWARE acrescenta novas funções ao BASIC, tais como Cópia e Movimentação de Linhas, Busca de Constantes ou Listagem Controlada de Programas, sendo uma ferramenta de trabalho de extrema utilidade.
Possibilita ainda o acionamento de comandos BASIC por intermédio de "Sintaxe Simplificada", assim como o acionamento de "programas BASIC de uma linha".
É TRANSPARENTE ao usuário do BASIC, que continua integralmente disponível, simultaneamente ao BIT-BASIC.
A característica de ERGONOMIA foi buscada para os seus comandos, objetivando um mínimo esforço físico e intelectual em sua utilização.

UMA PONTE PARA O ASSEMBLER

O LIVRO tem DOIS OBJETIVOS.

O primeiro é mostrar as técnicas de interceptação e comunicação com o BASIC, via ASSEMBLER-Z80, abrindo ao usuário a possibilidade de modificar e adaptar as suas funções.
Isto é feito a partir da própria DESCRIÇÃO DETALHADA DO BIT-BASIC, que assim é apresentado como um software ABERTO AOS USUÁRIOS.

O segundo objetivo é constituir-se em UMA PONTE PARA O ASSEMBLER, para os usuários de BASIC não familiarizados com esta linguagem.
Partindo apenas do conhecimento de BASIC, e utilizando este conhecimento como base para comparações, o leitor será capaz de conversar com o MSX na linguagem do Z-80, o seu processador principal.

A associação do LIVRO com o SOFTWARE teve como fundamento a busca da MULTIPLICAÇÃO DO CONHECIMENTO, e espero que possa contribuir para que os usuários do MSX, este versátil e singular equipamento, utilizem cada vez mais e melhor os seus recursos.

O AUTOR

INTRODUÇÃO

A abordagem utilizada para explicar o funcionamento do Z-80 e do ASSEMBLER é a mais objetiva possível, suficiente para a "Iniciação" nesta linguagem, procurando sempre explicações por intermédio de exemplos, com referências a conceitos já conhecidos de BASIC.
Com isto é possível um aprendizado "Prático", com utilização imediata no desenvolvimento e análise "Daquelas" rotinas ASSEMBLER que você sempre desejou utilizar e entender.
Os novos conhecimentos adquiridos, ou aqueles já possuídos, podem ser então consolidados pelo estudo do BIT-BASIC, o que proporcionará também um entendimento do funcionamento interno do BASIC.

O controle sobre o processamento é dinamicamente compartilhado pelo BASIC e pelo BIT-BASIC, estando ambos ativos todo o tempo.
Tudo se passa como se o BASIC tivesse sido "Modificado" e os novos comandos a ele pertencessem.

O BIT-BASIC procura utilizar sempre que possível as funções do BIOS e do BASIC, com o que é obtido um alto grau de SIMPLICIDADE, CONFIABILIDADE e COMPATIBILIDADE.
A "Administração da Tela" e a "Inclusão/Modificação/Exclusão de Linhas BASIC" são exemplos disto.

São deixados BYTES não utilizados entre as rotinas do programa BIT-BASIC para faciliar possíveis adaptações que você deseje efetuar (você poderá fazer estas adaptações diretamente em "Linguagem de Máquina", se elas forem "Simples").

O conteúdo dos BYTES estará sempre representado na forma HEXADECIMAL equivalente, o que será indicado pelo caráter "#" precedendo a sequência de dígitos Hexadecimais (por exemplo, #3A, #4CE3).

Os Microcomputadores MSX HOTBIT e EXPERT possuem algumas diferenças, o que se reflete em diferenças correspondentes no BIT-BASIC para que rode em uma ou outra máquina.
Este é o caso da disposição diferente das teclas no Teclado e também da disposição diferente dos bancos de memória RAM.

Os termos técnicos em Inglês estão sempre traduzidos para os seus significados equivalentes em Português, para facilidade de entendimento.

CAPÍTULO I

INTERCEPTANDO O BASIC

01 - ROM e RAM, BIOS e BASIC.

O seu microcomputador MSX possui módulos de memória permanente, cujo conteúdo permanece intacto mesmo com o aparelho desligado.
Este tipo de memória é denominado ROM (Read Only Memory - Memória Unicamente de Leitura), e seu conteúdo não pode ser alterado nem mesmo com os recursos disponíveis no seu próprio equipamento. Assim, não há maneira de modificarmos os programas gravados em memória ROM.

Nestes bancos de memória estão gravados dois programas básicos para o funcionamento do seu microcomputador.

O primeiro deles é o BIOS (Basic Input Output System - Sistema Básico de Entrada e Saída), constituído por um conjunto de rotinas cujas principais funções são dar início ao funcionamento da máquina e controlar as operações básicas de entrada e saída dos dados, de/para os periféricos conectados (Teclado, Tela, Gravador, Diskette, Etc.).

O segundo programa gravado na memória ROM é o Interpretador BASIC (Begginner's All-purpose Symbolic Instruction Code - Código Simbólico de Instruções de Finalidade-geral para Iniciantes), cujo propósito essencial é facilitar o uso dos poderosos recursos de processamento digital, gráfico e sonoro disponíveis no seu MSX.
É ele que "interpreta" o "Programa BASIC" por você escrito, controlando sua execução.
Apesar da legenda "Para Iniciantes", cunhada à época da definição de seus primeiros objetivos, o BASIC possui todos os recursos necessários para ser empregado em ampla gama de aplicações avançadas, para várias áreas de atividades profissionais e de lazer.

Quando você liga o microcomputador é automaticamente acionado o BIOS, que em seguida passa o controle para o Interpretador BASIC (que, por sua vez, irá utilizar rotinas do BIOS).
Ambos os programas estão armazenados em memória do tipo ROM, e é por este motivo que não é necessário qualquer dispositivo externo para dar início ao funcionamento da máquina.

O seu MSX é ainda equipado com módulos de memória do tipo RAM (Random Access Memory - Memória de Acesso Aleatório) que pode ser alterada e mantida sob controle do microcomputador. É aqui que são armazenados os programas e comandos escritos pelo usuário ou carregados a partir de meios externos.

O microprocessador Z-80 pode operar com endereços de memória entre #0000 e #FFFF, e a memória ROM ocupa as posições compreendidas entre #0000 e #7FFF.

02 — GANCHOS

COMO MODIFICAR OS PROGRAMAS RESIDENTES EM MEMÓRIA ROM ?

Ainda que o armazenamento neste tipo de memória apresente vantagens importantes, algumas vezes pode trazer dificuldades para o usuário. Tal é a situação em que precisamos alterar o funcionamento de programas nela residentes.

Prevendo esta necessidade, as rotinas BIOS e o Interpretador BASIC efetuam desvios, em pontos estratégicos, para endereços pertencentes à memória RAM. Nestes endereços são colocadas, no momento de inicialização da máquina, instruções de retorno imediato à ROM, com o que não é produzida qualquer alteração funcional nos programas originais.

Com esta técnica, conhecida como GANCHO (HOOK), abre-se ao usuário a possibilidade de alterar o conteúdo dos citados endereços da RAM, sendo possível introduzir modificações funcionais nas rotinas do BIOS e do Interpretador BASIC, ainda que a codificação original permaneça intacta.

Os endereços de memória onde estão armazenadas as instruções de desvio vão de #FD9A até #FFCA (cinco Bytes para cada GANCHO).

03 — INTERCEPTANDO O BASIC.

MODOS DE COMUNICAÇÃO COM O USUÁRIO.

O Interpretador BASIC possui dois modos de execução de Instruções.

No modo DIRETO, cada linha digitada no teclado é interpretada e executada imediatamente após o pressionamento da tecla RETURN (este modo é utilizado quando a linha não inicia por dígitos numéricos).

No modo INDIRETO o Interpretador BASIC analisa e controla a execução de um conjunto de instruções previamente armazenadas na memória RAM pelo usuário. Neste modo as linhas introduzidas no modo DIRETO, iniciando cada uma por um número compreendido entre 0000 e 65529. Este conjunto de linhas constitui-se em um "PROGRAMA BASIC".

ROTINA E PONTO DE INTERCEPTAÇÃO.

Para cada linha entrada no modo DIRETO o Interpretador BASIC passa sempre pelas instruções armazenadas na ROM a partir do endereço #4134. Este é o endereço inicial da rotina chamada MAIN-ENTRY (ENTRADA-PRINCIPAL), cuja primeira instrução é uma chamada (GANCHO) para o endereço #FF0C da RAM.

Se desejássemos interceptar o Interpretador BASIC neste ponto para, por exemplo, acionarmos o alarme ("BEEP") antes da aceitação de cada linha, poderíamos alterar o conteúdo da RAM a partir de #FF0C.

Experimente aplicar os seguintes "POKES" (nesta ordem), e "Ouça" o resultado (atenção - não faça este teste com o BIT-BASIC "funcionando") :

```
POKE &HFF0D,&HC0
POKE &HFF0E,&H00
POKE &HFF0C,&HCD
```

Isto faz com que, a cada passagem do Interpretador BASIC pela rotina MAIN-ENTRY, seja executada a instrução ASSEMBLER:

CALL #00C0

correspondente à instrução do Z-80:

#CD #C0 #00 .

cuja função é "Chamar" a rotina "BEEP" do BIOS, instalada a partir do endereço #00C0.
A instrução seguinte, já presente na memória, é

RETURN (#C9)

com a qual é efetuado o RETORNO à "ROTINA PRINCIPAL" do BASIC, após a execução da rotina "BEEP".

Para "Desligar" o desvio, faça "POKE&HFF0C,&HC9".

Não é este, porém, o ponto utilizado para interceptação do Interpretador BASIC.
Mais adiante, em #4164, é efetuada chamada ao endereço #00AE, onde existe uma rotina do BIOS denominada PINLIN, cuja função é "Receber uma Linha Digitada até o Pressionamento da Tecla RETURN ou das Teclas CONTROL+STOP" (em #00AE há um desvio para #23BF que é o endereço onde se inicia de fato esta rotina).

A primeira instrução de PINLIN é uma chamada (GANCHO) ao endereço #FDDB da RAM (CALL #FDDB = #CD #DB #FD). Em #FDDB, substituímos as instruções de retorno à ROM (RETURN = #C9) por um desvio para outro endereço da RAM, onde colocamos a rotina DESVIO do BIT-BASIC (#FFD9).

Este desvio é implementado pela instrução:

JUMP #FFD9 (#C3 #D9 #FF)

armazenada a partir do endereço #FDDB.

CAPÍTULO II

O BIT - BASIC

01 - ROTINA DESVIO

01.1 - OBJETIVO

O objetivo da rotina DESVIO do BIT-BASIC é interpretar cada linha digitada pelo usuário MSX no modo DIRETO e decidir se esta deve ser processada pelo Interpretador BASIC ou pelo BIT-BASIC.
Esta rotina executa também a tarefa de "Comunicação" (Transferência de Controle) do BIT-BASIC com o BIOS/BASIC.

01.2 - CARACTERÍSTICAS

O endereço escolhido para o início da rotina DESVIO é #FFD9, pertencente ao espaço livre existente após a área de GANCHOS entre os endereços #FFCA até #FFFE (endereços não utilizados pelo BIOS/BASIC).
A técnica utilizada para analisar o conteúdo da linha ANTES do Interpretador BASIC é acionar "em seu lugar" a rotina PINLIN, que devolve então o controle para a rotina DESVIO do BIT-BASIC.
A utilização desta técnica se deve ao fato de não existir um GANCHO instalado no final da rotina PINLIN.

01.3 - ROTINA ASSEMBLER

```
0010          ORG   #FFD9
0020 ;
0030 DESVIO:  POP   BC
0040          CALL  #23C2
0050          RET   C
0060          PUSH  HL
0070          RST   #10
0080          LD    HL,DV09
0090          LD    BC,#0004
0100          AND   A
0110          CPIR
0120          POP   HL
0130          RET   NZ
0140 DV01:    POP   BC
0150          LD    BC,INICIO
0160          PUSH  BC
0170 DV02:    LD    A,#XX        (#FC/HOTBIT, #A0/EXPERT)
0180 DV03:    OUT   (#A8),A
0190          RET
0200 DV09:    DEFB #3C,#5B,#3D,#2E  (HOTBIT)
                   #2F,#5B,#3D,#2E  (EXPERT)
```

0010 Instrui o compilador ASSEMBLER para colocar as próximas instruções do BIT-BASIC a partir do endereço #FFD9.

0030 Retira da PILHA DO SISTEMA o endereço aí colocado pela primeira instrução da rotina PINLIN (CALL #FDDB).
Com isto o próximo endereço da PILHA DO SISTEMA passa a apontar para a instrução imediatamente seguinte àquela que efetuou a chamada à rotina PINLIN.

0040 Chama a rotina PINLIN do BIOS na mesma situação em que ela seria chamada pela rotina "MAIN-ENTRY" do BASIC.
(#23C2 é o endereço de sua segunda instrução, pois a primeira é a chamada ao GANCHO).
Com isto é possível obter o controle APÓS a digitação da linha e ANTES de sua interpretação pelo BASIC, analisando o seu conteúdo e retornando ao processamento normal, ou continuando o processamento sob controle do BIT-BASIC.

0050 Retorna ao BASIC se as teclas CONTROL+STOP foram acionadas durante a execução da rotina PINLIN (esta rotina liga o "Indicador de Carry-Status" caso isso ocorra).
O retorno será efetuado para a instrução imediatamente seguinte àquela que "chamou" PINLIN (veja Instrução 0030), com o que o processamento continua normalmente, como se o BIT-BASIC não existisse.

0060 Salva REG-HL na PILHA DO SISTEMA, o qual contém o endereço de início do BUFFER (Área de Trabalho), onde é colocada a linha digitada.
Ao retornar da rotina PINLIN o REG-HL contém o endereço (menos 1) deste BUFFER, a partir do qual o BASIC efetua o processamento da linha digitada (o BIT-BASIC também processa a linha a partir deste endereço).
O BUFFER inicia em #F55E, e o REG-HL contém #F55D.
É conveniente observar que antes da linha ser colocada no BUFFER citado os códigos digitados pelo usuário são colocados em outra área de trabalho intermediária, chamada KEYCODE-BUFFER (BUFFER do Teclado), de onde os caracteres são retirados pela rotina PINLIN. Esta área será utilizada para outras funções do BIT-BASIC posteriormente descritas.

0070 Obtém o próximo (Primeiro) caráter da linha digitada e o coloca no REG-A.
A instrução RST #10 (#D7) efetua uma "Chamada" (equivalente a um CALL) ao endereço #0010 da ROM, onde está instalada a rotina CHRGTR do BIOS, cuja função é colocar no REG-A o "Próximo Caráter" armazenado no BUFFER de entrada (#F55E).
Esta rotina posiciona o indicador CY = 1 caso o caráter seja numérico, e o indicador Z = 1 se for encontrado o "Fim da Linha" (Caráter = #00).
O REG-HL contém o endereço do caráter recuperado.
A primeira instrução da rotina CHRGTR é uma chamada (GANCHO) para o endereço #FF48 da RAM (a rotina CHRGTR inicia de fato no endereço #4666 da ROM, pois em #0010 há um desvio para #2686, e neste endereço um desvio para #4666).

0080 Carrega no REG-HL o endereço da tabela do BIT-BASIC de nome DV09, que inicia no endereço #FFF5 e possui 04 caracteres.

0090 Carrega no REG-BC o valor #0004 (tamanho da tabela DV09).
0100 "Força" o valor "Zero" (desligado) no indicador de Carry-Status (CY), para que o BASIC não "pense" que as teclas CONTROL+STOP foram pressionadas, caso seja efetuado o retorno na instrução 0130.
(O conteúdo do REG-A não é alterado por esta instrução.)
0110 Procura um caráter igual ao contido no REG-A nos quatro BYTES (REG-BC = #04) a partir do endereço #FFF5 (REG-HL = tabela DV09).
Posiciona o indicador Z = 1 (Ligado) caso o caráter procurado seja encontrado.
(O indicador CY não é alterado por esta instrução.)
0120 Retorna ao REG-HL o endereço do BUFFER (#F55D), armazenado na PILHA DO SISTEMA pela instrução da linha 0060.
0130 Retorna ao BASIC e continua o processamento normal da linha, caso o seu primeiro caráter não seja "<, [, =, ." (HOT-BIT) ou "/, [, =, ." (EXPERT).
Caso o primeiro caráter da linha digitada pelo usuário seja um destes, esta linha é identificada como um "comando BIT-BASIC" sendo o controle passado para a instrução 0140, sem devolução ao Interpretador BASIC.
0140 Retira da PILHA DO SISTEMA o endereço de retorno à chamada original da rotina PINLIN pelo Interpretador BASIC. Isto é feito porque o BIT-BASIC fará o retorno para outros endereços do BASIC diferentes deste, dependendo do tipo de comando e das condições encontradas.
0150 Carrega no REG-BC o endereço da rotina INICIO do BIT-BASIC (#7000).
0160 Coloca na PILHA DO SISTEMA o endereço da rotina INICIO.
0170 Coloca no REG-A o valor #FC (HOT-BIT) ou #A0 (EXPERT).
0180 Habilita (Torna "ativo") o bloco de memória RAM onde está armazenado o BIT-BASIC.
Este módulo de memória ocupa os endereços de #4000 a #7FFF, que são endereços ocupados PELO PRÓPRIO Interpretador BASIC. (Como isto é possível??? Você verá em seguida, no ítem II-2.)
O valor #FC ou #A0 contido no REG-A é que determina a "Combinação" dos módulos de memória desejamos ativar. O valor #A8 indica à PPI (veja item II-2) "O QUE" desejamos mudar, no caso a configuração dos blocos de memória.
0190 A instrução RET equivale a um "Desvio Incondicional" para o endereço armazenado no topo da PILHA DO SISTEMA. Como este endereço é o da rotina INICIO (aí colocado pela instrução 0160 - PUSH HL), o Z-80 desvia para esta rotina que efetua os primeiros procedimentos do BIT-BASIC.

**** As instruções das linhas 00160 a 00190 são chamadas por diversas outras rotinas do BIT-BASIC, para efetuar transferências de endereços/controle com mudança de SLOT (bloco de memória ROM ou RAM).

02 - ROTINA INICIO

02.1 - OBJETIVO

O objetivo da rotina INICIO é dirigir o BIT-BASIC para a rotina apropriada, dependendo do tipo de comando recebido.

02.2 - CARACTERÍSTICAS

São os seguintes os comandos e funções correspondentes a cada caráter que, colocado na primeira posição da linha, faz com que ela seja tratada como um comando para o BIT-BASIC.

HOTBIT

```
< = #3C  Rotinas para avançar/listar página.
[ = #5B  Rotina para retroceder linhas/listar página.
= = #3D  Rotina para pesquisar constante.
. = #2E  Rotinas diversas, conforme próximos caracteres.
```

EXPERT

```
/ = #2F  Rotinas para avançar/listar página.
[ = #5B  Rotina para retroceder linhas/listar página.
= = #3D  Rotina para pesquisar constante.
. = #2E  Rotinas diversas, conforme próximos caracteres.
```

A escolha dos caracteres está associada à facilidade de acesso às teclas correspondentes (teclas mais próximas à tecla RETURN).

A existência do BIT-BASIC é transparente ao usuário MSX caso o primeiro caráter da linha não seja um dos anteriormente descritos, ou seja: tudo se passa como se ele não existisse, permanecendo todas as funções do Interpretador BASIC plenamente ativas.

À exceção da rotina de comunicação ("Interface") do BASIC com o BIT-BASIC (rotina DESVIO, já descrita), todo o restante deste programa fica instalado em memória RAM, a partir do endereço #7000, para evitar a redução do espaço de memória destinado aos "Programas BASIC", que ficam acima do endereço #8000 na RAM.

Agora, uma pergunta :

COMO É POSSÍVEL ARMAZENAR UM PROGRAMA A PARTIR DO ENDEREÇO #7000 SE TAL ENDEREÇO PERTENCE À MEMÓRIA DO TIPO ROM ???

Na verdade, o microcomputador MSX possui também "bancos" de memória RAM instalados "paralelamente" à memória ROM, ocupando os mesmos endereços, para ser utilizado alternativamente a esta última (num dado instante somente uma delas pode estar "ativa").
Assim podemos fazer com que o microprocessador Z-80, ao acessar um endereço compreendido entre #0000 e #7FFF, esteja se referenciando à memória ROM (onde estão os programas BIOS e BASIC), ou à memória RAM "Paralela" (onde podem ser colocados programas do usuário).
A operação de "Comutação" ou "Chaveamento" para ativar um ou outro bloco de memória é efetuada por um microprocessador especial, contido no seu MSX, denominado PPI (Programmable Peripheral Interface - Interface de Periféricos Programável), que pode ser acionado a partir do próprio Z-80 por intermédio de instruções especiais para esta finalidade (a instrução 00180 da rotina DESVIO executa a operação de "trocar" a memória ROM para RAM entre os endereços #4000 e #7FFF).
A rotina INICIO fica armazenada de #7000 a #7046 na memória RAM "Paralela" à memória ROM ocupada pelo Interpretador BASIC.

02.3 - ROTINA ASSEMBLER

```
0220            ORG  #7000
0230 ;
0240 INICIO:    LD   A,(T19)
0250            CP   #C3
0260            JP   Z,CML02
0270            PUSH HL
0280 A01:       CALL RST10
0290            CALL MINUSC
0300            LD   (HL),A
0310            CP   #00
0320            JR   NZ,A01
0330            POP  HL
0340            CALL RST10
0350            CP   #XX          (#3C/HOTBIT -  #2F/EXPERT)
0360            JP   Z,LISTAPG
0370            CP   #5B
0380            JP   Z,VOLTLIN
0390            CP   #3D
0400            JP   Z,VERCTE
0410            CALL RST10
0420            JP   Z,RET03
0430            JP   C,LISTLIN
0440            CP   #7A
0450            JP   Z,NOVOBAS
0460            CP   #63
0470            JP   Z,COPMOV
0480            CP   #6D
0490            JP   Z,COPMOV
0500            CP   #3D
0510            JP   Z,ENCERRA
0520            JP   SINTAXE
```

0220 Instrui o compilador ASSEMBLER a colocar o programa "Z-80" a partir do endereço #7000.

0240 Carrega no REG-A o conteúdo da "Área de Trabalho" de nome "T19" (um BYTE), cujo propósito é indicar que a rotina COPMOV, destinada a mover e copiar linhas BASIC, está "Sendo Executada" (veja detalhes na descrição desta rotina).

0250 Compara o conteúdo do REG-A com #C3 (caso afirmativo, posiciona o Indicador Z = 1).

0260 Desvia para a rotina CML02 caso o REG-A contenha #C3 (Indicador z "Ligado"- Cópia de Linha "Em Progresso").

**** As linhas 0270 a 0330 têm como objetivo converter para "minúsculos" todos os caracteres alfabéticos que tenham sido digitados como "maiúsculos", com a finalidade de facilitar o processamento da linha pelo BIT-BASIC.

0270 Salva na PILHA o REG-HL (#F55D).

0280 Obtém, no REG-A, o próximo caráter da linha digitada.

0290 Converte caráter alfabético do REG-A para "minúsculo".

0300 Recoloca o caráter convertido no BUFFER (na posição apontada por REG-HL).

0310 Verifica se o BYTE obtido contém #00. Este valor é colocado pela própria rotina PINLIN para indicar a condição de "Fim da Linha", após o último caráter digitado (no acionamento da tecla RETURN).

0320 Caso a condição de "Fim de Linha" não tenha sido atingida, retorna para a instrução ASSEMBLER de nome "A01" (linha 0280), para obter e converter o próximo caráter.

0330 Após a conversão de todos os caracteres alfabéticos para "minúsculos", restaura em REG-HL o endereço da primeira posição do BUFFER (#F55D), antes de iniciar o processamento da linha.

0340 Obtém, em REG-A, o primeiro caráter da linha digitada.

0350 Compara o caráter contido em REG-A com #3C = "<" (HOT-BIT) ou #2F = "/" (EXPERT).

0360 Se caráter "<" ou "/", desvia para rotina LISTAPG (Lista Página).

0370 Compara REG-A com #5B = "[".

0380 Se "[", desvia para VOLTLIN (Retrocede Linha e Lista Página).

0390 Compara REG-A com #3D = "=".

0400 Se "=" desvia para VERCTE (Procura Constante).
Se não, assume que é "." (caracteres já foram pesquisados na rotina DESVIO, linha 0110).

0410 Obtém próximo caráter da linha digitada para identificar a função do comando iniciado por ".".

0420 Desvia para a rotina RET02 caso tenha sido atingida a condição de "Fim de Linha" (o Indicador Z é posicionado pela rotina RST10 nesta condição).
A rotina RET02 devolve o controle do processamento ao Interpretador BASIC, que tratará a linha como um "ERRO DE SINTAXE" (experimente digitar "." e acionar RETURN e veja o que acontece).

0430 Desvia para a rotina LISTLIN (equivalente ao comando LIST do BASIC) caso o Indicador CY tenha sido "Ligado" pela rotina RST10 (isto significa que o primeiro dígito após o "." é numérico).
Caso o Indicador CY não esteja ligado, continua o processamento na próxima linha (0440).
0440 Compara REG-A com #7A = "z".
0450 Se "z" desvia para a rotina NOVOBAS (posiciona variáveis para inclusão de novo programa BASIC após o programa corrente).
0460 Compara REG-A com #63 = "c".
0470 Se "c" desvia para a rotina COPMOV (Copia/Move Linhas).
0480 Compara REG-A com #6D = "m".
0490 Se "m" desvia para a rotina COPMOV (Copia/Move Linhas).
0500 Compara REG-A com #3D = "=".
0510 Se "=" desvia para a rotina ENCERRA (desativa o BIT-BASIC).
0520 Se REG-A diferente de qualquer dos anteriores, desvia para a rotina SINTAXE (Comandos/Programas com Sintaxe Simplificada).

03 - ROTINA RST10

03.1 - OBJETIVO

O objetivo desta rotina é "obter o próximo caráter da linha digitada" colocando-o no Registrador A.
O resultado é equivalente ao da rotina CHRGTR (RST #10) do BASIC, já descrita no item II-01.3, instrução 0070.

03.2 - CARACTERÍSTICAS

O motivo pelo qual foi codificada uma nova é que a rotina CHRGTR está instalada na memória ROM que não está "ativa" durante a execução do BIT-BASIC, com o que seria necessário "Chavear/Deschavear" os blocos de memória a cada novo caráter. Não seria uma boa técnica executar um grande número de vezes os procedimentos relativamente "trabalhosos" de chaveamento.

Identicamente à rotina CHRGTR, a rotina RST10 "liga" o Indicador CY caso o dígito obtido seja numérico, e "liga" o Indicador Z caso tenha sido atingida a condição de "Fim de Linha" (Caráter #00).
O teste da condição de "numérico" é efetuado verificando se o conteúdo do REG-A está compreendido entre os valores #30 e #39, que correspondem aos caracteres numéricos de "0" até "9" na codificação ASCII (American Standard Code for Information Interchange - Padrão Americano de Codificação para Troca de Informações).

03.3 - CONDIÇÕES DE ENTRADA

REG-HL - Endereço anterior ao do caráter a ser obtido.

03.4 - CONDIÇÕES DE SAIDA

```
REG-HL - Endereço do caráter obtido (BUFFER).
REG-A  - Caráter obtido.
Z      - Fim da Linha (Z=1).
C,NZ   - Caráter Numérico (CY=1, Z=0).
NC,NZ  - Caráter Não-Numérico (CY=0, Z=0).
```

03.5 - ROTINA ASSEMBLER

```
0550 RST10:    INC  HL
0560           LD   A,(HL)
0570           CP   #00
0580           RET  Z
0590           CP   #30
0600           CCF
0610           RET  NC
0620           CP   #3A
0630           RET  C
0640           OR   A
0650           RET
```

0050 Soma "1" no endereço contido no REG-HL (o par de registradores HL passa a apontar para o "próximo" caráter, o qual desejamos obter).
(Por este motivo, para obtermos o primeiro caráter da linha, devemos colocar no REG-HL o "Endereço do BUFFER menos 1".)

0560 Coloca no REG-A o conteúdo do BYTE apontado pelo endereço contido no REG-HL.

0570 Compara o conteúdo do REG-A com #00, que é o código utilizado para indicar a condição de "Fim de Linha".

0580 Caso a condição de fim de linha tenha sido atingida, esta Instrução comanda ao Z-80 o Retorno à instrução imediatamente seguinte àquela que efetuou a chamada à rotina RST10 (com o "Indicador Z" ligado).

0590 Compara REG-A com #30. O conteúdo do REG-A não é modificado, porém os indicadores CY e Z são posicionados como se tivesse sido subtraído do REG-A o valor #30 (REG-A - #30). Isto resulta em :

REG-A < #30	C,NZ	(CY=1 - Resultado Negativo). (Z=0 - Resultado Diferente de Zero).
REG-A = #30	NC,Z	(CY=0 - Resultado Não-Negativo). (Z=1 - Resultado Igual a Zero).
REG-A > #30	NC,NZ	(CY=0 - Resultado Positivo). (Z=0 - Resultado Diferente de Zero).

0600 CCF = Complement Carry Flag - Complementa o Indicador de Transporte (CY).
"Complementa" ("Inverte") o valor do Indicador CY (se 1 transforma em 0, e vice-versa - Liga/Desliga).
Esta operação é feita para atender às CONDIÇÕES DE SAÍDA já mostradas.

0610 RETORNA caso o Indicador CY esteja "Desligado" ("0").
A combinação das instruções 0590 e 0600 faz com que esta condição seja verdadeira se REG-A < #30.

0620 Compara o REG-A com #3A.
De forma semelhante ao explicado para a instrução 0590, são posicionados os indicadores CY e Z.

0630 RETORNA caso o Indicador CY esteja "Ligado" (o Indicador Z estará "Desligado" neste caso).
Esta condição será verdadeira se REG-A < #39.

0640 O objetivo desta instrução é "forçar" o posicionamento dos indicadores CY e Z, para indicar que o caráter contido no REG-A é "Não Numérico" e é "Diferente de #00".
Como estamos comparando REG-A com ele próprio, o seu conteúdo permanecerá o mesmo, porém os Indicadores de Estado ficarão : NC,Z (CY = 0, Z=1).
(O Indicador CY sempre é posicionado em "0" por esta instrução, e o BYTE resultante é diferente de #00 pois esta condição já foi testada na instrução 0570.)

0650 RETORNA à instrução imediatamente seguinte àquela que efetuou a chamada à rotina RST10.

04 — ROTINA MINUSC

04.1 - OBJETIVO

O objetivo desta rotina é converter para "Minúsculo" o caráter contido no REG-A, no caso deste caráter ser "Alfabético" e "Maiúsculo".

04.2 - CARACTERÍSTICAS

O MSX segue o padrão "ASCII", no qual os caracteres "Alfabéticos Maiúsculos" correspondem aos valores hexadecimais compreendidos entre #41 ("A") e #5A ("Z"), e os caracteres "Alfabéticos Minúsculos" correspondem aos valores hexadecimais comprendidos entre #61 ("a") e #7A ("z").
Portanto, podemos observar que entre cada caráter alfabético maiúsculo e o seu caráter minúsculo correspondente existe uma "diferença" de #20.
Assim, se "Somarmos #20" ao "Valor Hexadecimal" que corresponde a um caráter "Maiúsculo", obteremos o "Valor Hexadecimal" do caráter "Minúsculo" correspondente.

Por exemplo :
A = #41
a = #41 + #20 = #61
Z = #5A
z = #5A + #20 = #7A

04.3 - CONDIÇÕES DE ENTRADA
REG-A - Caráter a ser convertido

04.4 - CONDIÇÕES DE SAÍDA
REG-A - Caráter já convertido

04.5 - ROTINA ASSEMBLER

```
0680 MINUSC:    CP   #41
0690            RET  C
0700            CP   #5B
0710            RET  NC
0720            ADD  A,#20
0730            RET
```

0680 Compara REG-A com #41 ("A").
0690 RETORNA se REG-A < #41 (caráter não é "Alfabético Maiúsculo").
0700 Compara REG-A com #5B (#5A = "Z").
0710 RETORNA se REG-A maior ou igual a #5B (caráter não é "Alfabético Maiúsculo").
0720 Soma #20 ao REG-A (transforma caráter Alfabético "Maiúsculo" em "Minúsculo").
0730 RETORNA à instrução imediatamente seguinte àquela que efetuou a chamada à rotina MINUSC.

05 - ROTINAS CALL/JUMP

05.1 - OBJETIVO

O objetivo da rotina CALL é efetuar "Chamadas" a rotinas do Interpretador BASIC, as quais não são diretamente acessíveis a partir do BIT-BASIC em razão de não estar "ativa" a memória ROM onde elas estão instaladas (neste momento, os endereços #4000 a #7FFF correspondem à memória RAM, ativada pela rotina DESVIO).

O objetivo da rotina JUMP é efetuar "Desvio" para endereços pertencentes ao Interpretador BASIC, sendo válidas as mesmas considerações efetuadas para a rotina CALL.

05.2 - CARACTERÍSTICAS

A codificação das duas rotinas está "associada" e ambas estão interligadas aos "Pontos de Entrada" ("Entry-Points") DV02 e DV03 da rotina DESVIO.

05.3 - CONDIÇÕES DE ENTRADA

(CALL) REG-BC - Endereço da rotina chamada
(CALL) PILHA DO SISTEMA - Endereço de retorno
(JUMP) REG-BC - Endereço da rotina de destino

05.5 - CONDIÇÕES DE SAÍDA

(CALL) Desvio para o endereço contido na PILHA DO SISTEMA
(JUMP) Não há retorno

05.6 - ROTINA ASSEMBLER

```
0760 CALL:     LD   DE,DV02
0770           PUSH DE
0780 JUMP:     PUSH BC
0790           LD   A,#XX          (#F0/HOTBIT - #A0/EXPERT)
0800           JP   DV03
```

0760 Carrega no REG-DE o endereço do "Ponto de Entrada" DV02 da rotina INICIO (#FFF0), que corresponde à instrução "LD A,#XX" da linha 0170.
0770 Armazena o valor contido em REG-DE (#FFF0) na PILHA DO SISTEMA.
0780 Armazena o valor contido em REG-BC na PILHA DO SISTEMA. (Antes de utilizarmos a rotina CALL ou a rotina JUMP, devemos colocar no REG-BC o endereço da rotina a ser chamada.)
0790 Carrega no REG-A o valor #F0 (HOTBIT) ou #A0 (EXPERT).
0800 Desvia para o ponto de entrada DV03 da rotina INICIO.

**** A seguir repetiremos o funcionamento das instruções da rotina INICIO utilizadas para concluir as funções CALL e JUMP.

0180 OUT (#A8),A - "Reativa" o bloco de memória ROM onde está o Interpretador BASIC.
0190 RET - A instrução Assembler "RET" (Z-80 = #C9) faz com que o "Programa Z-80" efetue um desvio para o último endereço armazenado na PILHA DO SISTEMA (passando a apontar para o próximo endereço da PILHA).
O último endereço que colocamos na PILHA foi aquele contido no REG-BC (veja linha 0780), que é exatamente o endereço inicial da rotina com a qual desejamos nos comunicar.
Caso estejamos passando por este ponto do programa em função de uma chamada à rotina CALL (veja Instruções 0760 e 0770), o endereço seguinte da PILHA é o endereço de DV02 (#FFF0).

**** Como toda rotina acionada via CALL "termina" sempre com uma instrução RET, sempre haverá um desvio para DV02 após ter sido executada a rotina chamada, quando serão então executadas as instruções das linhas 0170 a 0190, cujo funcionamento nesta situação repetimos a seguir :

0170 Carrega em REG-A o valor #FC/#A8.
0180 Reativa o bloco de memória RAM onde está instalado o BIT-BASIC.
0190 Desvia para o último endereço armazenado na PILHA.
A "chamada" à rotina CALL do BIT-BASIC é sempre efetuada por uma instrução "CALL" de ASSEMBLER.
Portanto, quando é executada esta instrução RET, é efetuado um desvio para a "Instrução Imediatamente Seguinte àquela que Efetuou a Chamada à Rotina CALL", com o que é completado o ciclo de chamada.
No caso da rotina JUMP, que é acionada por uma instrução "JUMP" do Z-80, nada é colocado na PILHA, já que não será efetuado retorno.

RESUMIDAMENTE :

1 - Acionamento da rotina CALL (em qualquer ponto do BIT-BASIC).
2 - Armazena endereços "De retorno" e "Da rotina chamada" na PILHA DO SISTEMA (rotinas CALL/JUMP).
3 - "Ativa" memória ROM onde está o BASIC e desvia para a rotina chamada (via rotina DESVIO).
4 - Executa a rotina chamada.
5 - "Reativa" a memória RAM onde está o BIT-BASIC (via rotina DESVIO).
6 - Retorna ao ponto de partida (via rotina DESVIO).

06 — ROTINA RETO1

06.1 - OBJETIVO

O objetivo desta rotina é retornar ao Interpretador BASIC "Passando uma Linha de Comando", como se ela tivesse sido digitada via teclado pelo usuário.

06.2 - CARACTERÍSTICAS

SIMULANDO UMA LINHA DE COMANDO PARA O BASIC, KEY-CODE BUFFER

Esta técnica é utilizada para fazermos com que o Interpretador BASIC execute instruções comandadas pelo programa BIT-BASIC (por exemplo, FILES, LOAD, LIST, etc.).
O retorno é efetuado ao BASIC para o início da rotina MAIN-ENTRY, num ponto "Anterior à Chamada da Rotina PINLIN".
Os comandos desejados são colocados em uma área denominada KEY-CODE-BUFFER ("ÁREA DE TRABALHO - PARA CÓDIGOS - TECLADOS"), ou simplesmente KEYBUFFER (BUFFER DO TECLADO), que é onde são colocados os caracteres correspondentes às teclas que você aciona em seu microcomputador.
Esta área, com o tamanho de 40 BYTES, está colocada entre os endereços #FBFO e #FC17 e é controlada por dois dados (GETPNT/PUTPNT) que são atualizados cada vez que um caráter é aí colocado ou retirado.

GETPNT E PUTPNT

O primeiro dado de controledo BUFFER DO TECLADO é o "Endereço do Primeiro Caráter Digitado", que fica armazenado nos endereços #F3FA/#F3FB e tem o nome de GETPNT (GET POINT - Ponto de Entrada).
O segundo dado é o "Endereço Após o Último Caráter Digitado", que fica armazenado no endereços #F3F8/#F3F9 e tem o nome de PUTPNT (PUT POINT - Ponto de Saída).

- Se GETPNT = PUTPNT, significa que o BUFFER está "Vazio".
- Se GETPNT # PUTPNT, significa que "há caracteres digitados e ainda não processados no BUFFER".
- A cada caráter entrado PUTPNT é incrementado de 1, e a cada caráter retirado GETPNT é incrementado de 1 (de forma "circular", de maneira que nunca são ultrapassados os limites do BUFFER e número máximo de caracteres armazenados seja 40).

A rotina PINLIN, cuja função é "Obter uma Linha Digitada até RETURN ou CONTROL+STOP (veja ítem 01), executa a tarefa de retirar do BUFFER os caracteres lá existentes, até que seja encontrado um caráter que represente o fato de ter sido pressionada a tecla RETURN (o caráter utilizado para isto é #0D) ou um caráter que represente o pressionamento das teclas CONTROL+STOP (o caráter utilizado é #03).

SIMULANDO UMA LINHA DE COMANDO

Para simularmos ao BASIC a existência de uma linha de comandos a ser processada, como se ela tivesse sido introduzida via teclado, o texto correspondente aos comandos desejados deve ser colocado no BUFFER DO TECLADO e as variáveis GETPNT e PUTPNT devem ser posicionadas respectivamente no "Primeiro" e "Último Caráter Mais 1" deste texto, desviando-se em seguida para o BASIC no ponto de chamada da rotina PINLIN, que fará a retirada da linha a partir do BUFFER e a entregará ao Interpretador BASIC para processamento.
ESTE É O PROCEDIMENTO EFETUADO PELA ROTINA RETO1, PORÉM COM UMA VARIAÇÃO QUE DESCREVEMOS EM SEGUIDA.
A linha que desejamos passar ao BASIC é colocada na Tela do monitor ou do aparelho de TV conectado ao microcomputador. Isto pode ser feito com a utilização da rotina CHPUT (CHARACTER PUT - Saída de Caráter - Envia um caractere ao vídeo - CALL #00A2), ou da rotina OUTDO (OUT DO - Executa Saída - Envia um Dado ao Último Dispositivo Referenciado - RST #18 = CALL #0018).
Em seguida, com o CURSOR posicionado sobre a linha colocada na Tela, utilizamos a técnica anteriormente descrita para simularmos à rotina PINLIN apenas e tão somente o pressionamento da tecla RETURN. Ou seja, colocamos no KEYBUFFER o caráter #0D (RETURN), fazemos a variável GETPNT apontar para este caráter, fazemos a variável PUTPNT apontar para o BYTE seguinte, e desviamos para a rotina PINLIN (é como se tivesse sido acionada a tecla RETURN sobre uma linha da Tela).

ADMINISTRAÇÃO DA TELA

Na verdade, existe um terceiro componente envolvido na captura de dados para processamento pelo BASIC que é a função encarregada da "Administração de Tela".
Quando existem várias linhas apresentadas, podemos movimentar o CURSOR livremente para qualquer posição e acionar a tecla RETURN em qualquer ponto.
Isto feito, a função de "Administração de Tela" irá procurar "Para Frente" e "Para Trás" para descobrir onde "Começa" e onde "Termina" a linha sobre a qual o CURSOR está posicionado. Em seguida, é então passada ao Interpretador BASIC esta linha, via KEYBUFFER e via PINLIN conforme já descrito.
Este modo de funcionamento, extremamente prático, no qual podemos correr livremente toda a tela, é denominado "FULL SCREEN - Tela Completa".

Na verdade, a linha que aparece na tela está armazenada em um bloco de memória RAM "Especial", destinada exclusivamente para este fim, denominada VRAM (VIRTUAL RAM - RAM Virtual). Ela é administrada por um outro microprocessador especial contido no seu MSX chamado VDP (VIDEO DISPLAY PROCESSOR - Processador de Apresentação em Vídeo), o qual também é acessível via Z-80, existindo diversas rotinas do BIOS já prontas para efetuar este acesso.

ROTINA RET01

A rotina RET01, portanto, simula o acionamento da tecla RETURN, com o CURSOR já posicionado sobre uma linha da tela que contém os comandos que desejamos executar, com o que esta linha será lida pela função de "Administração de Tela" e passada ao BASIC para processamento, via KEYBUFFER e rotina PINLIN.

Esta rotina possui ainda um outro Ponto de Entrada (Entry-Point) denominado RET02 que serve para simular o acionamento de "outras teclas de controle" sobre a linha BASIC existente na tela.
Por exemplo, simular o pressionamento das teclas "CTRL + U", que executa a função de "Apagar Toda a Linha Sobre a Qual está o CURSOR", utilizada pela rotina do BIT-BASIC que Copia e Move linhas (COPMOV).

06.3 - CONDIÇÕES DE ENTRADA
CURSOR posicionado na Tela sobre a linha desejada

06.4 - ROTINA ASSEMBLER

```
0830 RET01:     LD    HL,#000D
0840 RET02:     LD    (#FBF0),HL
0850            LD    HL,#FBF0
0860            LD    (#F3FA),HL
0870            INC   HL
0880            LD    (#F3F8),HL
0890            LD    BC,#4137
0900            JR    JUMP
```

0830 Carrega #000D em REG-HL (#0D indica para PINLIN o pressionamento da tecla RETURN - Veja APENDICE 06).
0840 Coloca em #FBF0, endereço inicial do KEYBUFFER, o valor contido no REG-HL (#000d).
0850 Carrega no REG-HL o valor #FBF0, que corresponde ao endereço do KEYBUFF onde colocamos o "Primeiro Caráter da Linha".
0860 Armazena o valor de REG-HL na variável GETPNT (#F3FA).
0870 Incrementa REG-HL de uma unidade (posição seguinte ao caráter #0D).
0880 Armazena o valor de REG-HL na variável PUTPNT (#F3F8).
0890 Carrega em REG-BC o valor #4137, que é o endereço da segunda instrução da ENTRADA PRINCIPAL (MAIN-ENTRY) do Interpretador BASIC (a primeira é uma chamada ao GANCHO de #FF0C).
0900 Desvia para a rotina JUMP, que fará a "Ativação" do bloco de memória onde está o Interpretador BASIC e fará o desvio para a ENTRADA PRINCIPAL, onde será efetuada a chamada à rotina PINLIN, e tudo acontecerá conforme já descrito.

07 - ROTINA RET03

07.1 - OBJETIVO

O objetivo desta rotina é passar ao BASIC uma linha de comandos para processamento, porém SEM APRESENTAR ESTA LINHA NA TELA de seu monitor ou TV, não sendo, portanto, de conhecimento do usuário (com a técnica utilizada na rotina RET01 a linha de comandos é apresentada na tela).

07.2 - CARACTERÍSTICAS

A rotina PINLIN retira os caracteres do KEYBUFFER e os coloca em outra "Área de Trabalho" (BUFFER) que inicia no endereço #F55E, a partir da qual o BASIC finalmente efetua o seu processamento (caracteres colocados exatamente como foram digitados).
O tamanho desta área de trabalho é de "255+3", e é por este motivo que não conseguimos passar ao Interpretador BASIC linhas maiores do que 255 posições.

A rotina RET03 coloca a linha de comandos desejada diretamente neste BUFFER, como se tivesse sido entrada via PINLIN (e demais procedimentos já descritos), e a entrega ao Interpretador BASIC no ponto "imediatamente após o ponto de chamada a PINLIN" para que seja processada (instrução "RST #10") cuja função é "Retirar do Buffer um Caráter da Linha Digitada", colocando-o em REG-A e posicionando os Indicadores de Estado CY e Z (esta instrução fica no endereço #4173 da ROM e é "pulado" o teste efetuado após a rotina PINLIN para verificar se foram pressionadas as teclas CONTROL+STOP).

07.3 - CONDIÇÕES DE ENTRADA

Linha a ser passada ao Interpretador BASIC armazenada no BUFFER (#F55E).
LINHA A SER PASSADA AO BASIC ARMAZENADA NO BUFFER (#F55E).

07.4 - ROTINA ASSEMBLER

```
0930 RET03:     LD    HL,#F55D
0940            LD    BC,#4173
0950            JP    JUMP
```

0930 Carrega no REG-HL o valor #F55D, que é o "Endereço do BUFFER DO BASIC menos 1".
0940 Carrega no REG-BC o valor #4173, que é o endereço do BASIC "Após a Chamada da Rotina PINLIN".
0950 Desvia para o endereço #4173 do BASIC, via rotina JUMP.

08 - ROTINA RET04

08.1 - OBJETIVO

O objetivo desta rotina é efetuar o retorno, em situação normal, ao ponto inicial da "ENTRADA PRINCIPAL" ("MAIN-ENTRY") do Interpretador BASIC, onde será aceita uma nova linha de comandos (via rotina PINLIN) a qual passará por todos os procedimentos de interceptação e verificação já descritos.

08.2 - CARACTERÍSTICAS

O retorno é efetuado para o endereço #4137, que é a segunda instrução da ROTINA PRINCIPAL (a primeira é uma chamada ao GANCHO de #FF0C).

08.3 - ROTINA ASSEMBLER

```
0980 RET04:     CALL #0156
0990            LD   BC,#4137
1000            JP   JUMP
```

0980 Efetua chamada à rotina do BIOS denominada KILBUF (KILL BUFFER - Zera Buffer), encarregada de "Limpar" o KEYBUFFER (BUFFER do Teclado), o que é feito tornando "Iguais" os valores de GETPNT e PUTPNT.

0990 Carrega no REG-BC o valor #4137, que corresponde ao endereço de início da ROTINA PRINCIPAL do Interpretador BASIC.

1000 Desvia para o endereço #4137, conforme procedimentos já descritos para a rotina JUMP.

09 - ROTINA ENCERRA

09.1 - OBJETIVO

O objetivo desta rotina é "encerrar" o BIT-BASIC, desativando os procedimentos de interceptação do Interpretador BASIC.

09.2 - CARACTERÍSTICAS

Conforme já descrito no ítem I-03, a interceptação é implementada introduzindo-se um "desvio" para o BIT-BASIC a partir do GANCHO instalado em #FDDB, que é chamado no início da rotina PINLIN.
Êste desvio é feito para a rotina DESVIO pela instrução :

JP #FFD9 (#C3 #D9 #FF)

colocada a partir do endereço #FDDB.

Para desativarmos este desvio basta substituirmos o código #C3 (JUMP) por #C9 (RET), com o que será restaurado o retorno normal ao ponto de chamada do GANCHO na ROM.
Com isto o BIT-BASIC passa a ficar "Inativo" (ainda que suas instruções permaneçam na memória).

Esta função foi prevista em razão de que podemos desejar rodar outro programa que ocupe endereços já utilizados pelo BIT-BASIC. Neste caso sua codificação seria "Modificada" e, ao retornarmos ao BASIC, poderíamos obter resultados inesperados já que haveria desvio para uma combinação Imprevisivel de instruções.
(Esta condição NÃO SE APLICA para programas BASIC pois estes sempre ocupam endereços acima de #8000, não utilizados pelo BIT-BASIC.)

09.3 - ROTINA ASSEMBLER

```
1030 ENCERRA:  LD   A,#C9
1040           LD   (#FDDB),A
1050           JR   RET04
```

1030 Carrega o valor #C9 no REG-A (Código Instrução RET do Z-80).
1040 Carrega #C9 (RET) no endereço #FDDB.
1050 Desvia para o início da ROTINA PRINCIPAL do BASIC, via rotina RET04, para a continuidade normal de sua atividade.

10 — ROTINA PARM

10.1 - OBJETIVO

O objetivo desta rotina é capturar um parâmetro numérico a partir da "Linha de Comando BIT-BASIC", converter este valor para "Binário" e colocar este valor no REG-DE.

10.2 - CARACTERÍSTICAS

Por exemplo, se a linha de comando para o BIT-BASIC contém os caracteres "<<1279" nas suas primeiras posições (o que significa que desejamos "Listar o programa a partir da linha 1279"), a rotina PARM fará a conversão do número 1279, armazenado em QUATRO BYTES no BUFFER (ASCII) para o "Número Binário" equivalente armazenado em DOIS BYTES.
- ASCII : 1279 = #31 #32 #37 #39 (um BYTE p/cada digito)
- BINÁRIO : 1279 = #04FF (dois Bytes).

10.3 - CONDIÇÕES DE ENTRADA

REG-HL - Endereço do primeiro caráter do parâmetro numérico

10.4 - CONDIÇÕES DE SAÍDA

REG-DE - Valor Binário do Parâmetro Numérico
REG-HL - Endereço do Último Caráter do Parâmetro Numérico
C - Parâmetro numérico inválido (maior que 65525)
NC - Parâmetro numérico válido
Z - Parâmetro é "Zero"
NZ - Parâmetro diferente de Zero

10.5 - ROTINA ASSEMBLER (PARM)

```
1080 PARM:    CALL NUMX
1090          DEC  HL
1100          RET  C
1110          LD   A,E
1120          OR   D
1130          RET
```

1080 Efetua chamada à rotina NUMX a seguir descrita, que efetua a captura do parâmetro numérico.
1090 Diminui REG-HL de uma unidade.
O objetivo deste procedimento é "Voltar atrás" o último caráter lido pela rotina NUMX, ficando REG-HL posicionado sobre o "Último Caráter do Parâmetro Numérico" obtido.
1100 RETORNA caso o parâmetro numérico não seja válido (o Indicador de Estado CY é "Ligado" pela rotina NUMX se esta condição ocorrer).
1110 Carrega o conteúdo de REG-E em REG-A.
1120 Compara o conteúdo de REG-A com o conteúdo de REG-D, na modalidade OR, coloca o resultado em REG-A e posiciona os Indicadores de Estado CY e Z para mostrar se o conteúdo de REG-DE é igual ou diferente de #0000.
1130 RETORNA ao ponto de onde foi chamada a rotina PARM.

10.6 - ROTINA ASSEMBLER (NUMX)

```
1160 NUMX:     LD   DE,#0000
1170 A04:      LD   BC,#0000
1180           CALL RST10
1190           RET  Z
1200           RET  NC
1210           PUSH HL
1220           SUB  #30
1230           LD   C,A
1240           LD   H,D
1250           LD   L,E
1260           ADD  HL,HL
1270           ADD  HL,HL
1280           ADD  HL,HL
1290           ADD  HL,DE
1300           ADD  HL,DE
1310           ADD  HL,BC
1320           EX   DE,HL
1330           LD   HL,#FFF5
1340           RST  #20
1350           POP  HL
1360           RET  C
1370           JP   A04
```

1160 Carrega no REG-DE o valor #0000 ("Inicializa" REG-DE para posteriormente somar os digitos numéricos).

1170 Carrega no REG-BC o valor #0000.

1180 Obtém caráter a partir do BUFFER.

1190 RETORNA para a Instrução 1090 caso tenha sido atingida a condição de "Fim de Linha".

1200 RETORNA para a Instrução da linha 1090 caso o dígito obtido não seja numérico (isto indica a condição de "Fim do Parâmetro Numérico").

1210 "Salva" na PILHA DO SISTEMA o endereço contido no REG-HL, pois este Registrador será utilizado em seguida para "calcular" o valor binário do número.

1220 Subtrai #30 do Acumulador (REG-A). Com isto passamos a ter no REG-A um conteúdo compreendido entre #00 e #09 (já que os caracteres numéricos ASCII correspondentes vão de #30 a #39).
Este conteúdo corresponde à representação "Binária" dos dígitos numéricos de 0 a 9.

1230 Carrega o conteúdo do REG-A no REG-C. Assim o REG-BC passa a conter o "Valor Binário" do último caráter obtido (#00XX pois o REG-B contém #00).

1240 Carrega no REG-H o conteúdo do REG-D.

1250 Carrega no REG-L o conteúdo do REG-E.
Com as instruções das linhas 1240/1250, REG-DE é "Copiado" para REG-HL (o REG-DE é utilizado nesta rotina para armazenar o "Sub-Total" dos dígitos numéricos já processados).

**** As instruções das linhas 1260 a 1310 fazem com que o "Sub-Total" (Binário) contido no REG-HL seja multiplicado por dez, e com que seja somado o valor binário do novo caráter numérico obtido (contido no REG-BC).
Com a repetição deste procedimento, ao final da rotina, teremos acumulado no REG-DE os valores binários dos diversos dígitos que compõem o parâmetro numérico, cada qual multiplicado pelo "fator de dez" correspondente à sua posição relativa (o caráter mais à direita multiplicado por 1, o seguinte à sua esquerda por 10, o seguinte por 100, e assim por diante).

1260 Soma o valor contido no REG-HL ao próprio REG-HL (equivale a multiplicar por dois o valor contido no REG-HL).
1270 Dobra o valor de REG-HL.
1280 Dobra o valor de REG-HL.
1290 Soma no REG-HL o valor inicial do próprio REG-HL, copiado para o REG-DE pelas instruções 1240/1250.
1300 Soma novamente REG-DE no REG-HL.
1310 Adiciona no REG-HL o valor binário do último caráter obtido.
1320 Troca os conteúdos de REG-HL e REG-DE.
Neste ponto REG-DE passa a conter o "Sub-Total" dos dígitos processados até o momento.
1330 Carrega em REG-HL o valor #FFF5 = 65525.
1340 Efetua chamada à rotina DCOMPR que compara o valor contido no REG-HL com o valor contido no REG-DE (se REG-DE > REG-HL o Indicador CY é ligado).
1350 Restaura no REG-HL o endereço salvo na PILHA pela instrução da linha 1210.
Esta instrução não modifica os Indicadores de Estado CY e Z, com o que continua mantido o resultado da comparação efetuada na linha 1340.
A restauração tem que ser efetuada neste ponto porque pode haver Retorno na instrução seguinte e, neste caso, a PILHA DO SISTEMA deve estar posicionada no mesmo ponto no qual estava quando a rotina NUMX foi chamada.
1360 RETORNA da rotina NUMX caso o Indicador CY tenha sido ligado pela instrução 1340, o que indica "Parâmetro Inválido" (total já somado no REG-HL é maior do que 65525 - #FFF5).
1370 Desvia para a Instrução 1170 (A04) para a obtenção e processamento do próximo caráter contido no BUFFER.

11 — ROTINA PXLIN

11.1 - OBJETIVO

O objetivo desta rotina é "Recuperar" a "Próxima Linha" de um "Programa BASIC", a partir do seu "Endereço de Início", comparando o "Número" desta linha com um número fornecido.

11.2 - CARACTERÍSTICAS

Esta rotina utiliza os Indicadores de Estado CY e Z para comunicar à rotina "Chamadora" as condições encontradas. O endereço de início da "Linha Seguinte à Linha Recuperada" também é fornecido, para que a operação possa ser repetida, tornando possível recuperar todas as linhas do programa.

O endereço onde inicia o "Programa BASIC" (endereço de início da primeira linha) é guardado em um "Campo" de trabalho do Interpretador BASIC denominado TXTTAB (TEXT TABLE - ÁREA DE TEXTO), que fica em #F676/#F677 (dois BYTES).
Quando desejamos a primeira linha do programa, devemos colocar em REG-DE o valor contido no campo TXTTAB (normalmente #8001 - veja exemplo a seguir).

Deve ser fornecido um "Número de Linha" à rotina PXLIN, e ao seu retorno os indicadores CY e Z serão posicionados para refletir a situação da "Linha Recuperada" em relação a este Número (Menor, Igual ou Maior).

ARMAZENAMENTO DE UM PROGRAMA BASIC

Vejamos como são armazenadas as linhas de um programa BASIC.

Cada uma delas é precedida por dois BYTES que guardam o "Endereço de Início da Próxima Linha", mais dois BYTES que guardam o "Número da Linha" (em valor "Binário").
Cada uma delas é "Encerrada" por um BYTE que contém #00.

A condição de "Fim de Programa" é indicada por uma linha cujos dois primeiros BYTES contém #0000 (se esta linha for a primeira indica que não há nenhum programa BASIC carregado).

O "Conteúdo" (TEXTO) da linha é guardado de forma CODIFICADA, segundo critérios administrados pelo BASIC, não sendo diretamente legíveis sem um trabalho prévio de "Decodificação" dos COMANDOS, FUNÇÕES e PARÂMETROS.
Esta codificação visa reduzir o espaço de memória ocupado, além de facilitar os procedimentos de execução do programa (por exemplo, a instrução PRINT é representada por apenas um BYTE que contém o "Código" correspondente "#91" - Estes códigos são conhecidos pelo nome de TOKENS).

Por exemplo, o programa BASIC :

```
10 PRINT "BBB"
20 BEEP
30 END
```

É armazenado da seguinte maneira (Digite este programa e examine a memória com o comando ".h8000,8019" do BIT-BASIC) :

Posição Inicial do "Programa BASIC"

#8000	#00	Primeira posição sempre contém #00

Linha Número 10

#8001	#0C #80	Endereço início próxima linha (#800C)
#8003	#0A #00	Número da linha (#000A = 10)
#8005	#91	Código (TOKEN) da instrução PRINT
#8006	#22	Caráter "Aspas" (")
#8007	#42 #42 #42	BBB
#800A	#22	"Aspas"
#800B	#00	Fim da linha 10

Linha Número 20

#800C	#12 #80	Endereço início próxima linha (#8012)
#800E	#14 #00	Número da linha (#0014 = 20)
#8010	#C0	Código instrução BEEP
#8011	#00	Fim da linha 20

Linha Número 30

#8012	#18 #80	Endereço início próxima linha (#8018)
#8014	#1E #00	Número da linha (#001E = 30)
#8016	#81	Código instrução END
#8017	#00	Fim da linha 30

Condição de "Fim de Programa"

#8018	#00 #00	#0000 nos dois BYTES seguintes à última linha indicam "Fim do Programa BASIC"

11.3 - CONDIÇÕES DE ENTRADA

REG-DE - Endereço de início da "Linha a Ser Recuperada"
T03 - Número de Linha a ser comparado

11.4- CONDIÇÕES DE SAÍDA

REG-BC - Endereço de início da "Linha Recuperada"
REG-HL - Número da Linha Recuperada
REG-DE - Endereço de início da "Linha Seguinte à Linha Recuperada"
C,Z - Fim de Programa atingido
NC,Z - Número Linha Recuperada = Número Pesquisado
NC,NZ - Número Linha Recuperada > Número Pesquisado
C,NZ - Número Linha Recuperada < Número Pesquisado

11.5 - ROTINA ASSEMBLER

```
1400 PXLIN:     PUSH DE
1410            POP  BC
1420            LD   (A05+1),DE
1430 A05:       LD   HL,(#0000)
1440            LD   A,H
1450            OR   L
1460            SCF
1470            RET  Z
1480            INC  DE
1490            INC  DE
1500            LD   (A06+2),DE
1510 A06:       LD   DE,(#0000)
1520            LD   HL,(T03)
1530            RST  #20
1540            EX   DE,HL
1550            LD   (A07+2),BC
1560 A07:       LD   DE,(#0000)
1570            RET  Z
1580            CCF
1590            RET
```

1400 Salva na PILHA DO SISTEMA o conteúdo de REG-DE (endereço de início Linha a ser Recuperada).

1410 Coloca no REG-BC o último valor armazenado na PILHA DO SISTEMA.
(As instruções 1400/1410 efetuam a transferência do valor do REG-DE para o REG-BC, utilizando a PILHA como meio intermediário.)

**** As instruções 1420 e 1430 servem para colocar no REG-HL o "CONTEÚDO DOS BYTES APONTADOS PELO REG-DE".
Isto é conseguido fazendo com que a Instrução 1420 "MODIFIQUE O OPERANDO DE ENDEREÇO" da instrução 1430.
Com esta técnica, o BIT-BASIC MODIFICA SEU PRÓPRIO CÓDIGO ASSEMBLER para alcançar o objetivo desejado.

1420 Carrega no endereço "A05+1" (#7134), que corresponde ao "Operando de Endereço" da Instrução da Linha 1430 (A05), o valor contido no REG-DE.

1430 Carrega no REG-HL os dois BYTES apontados pelo operando de endereço desta instrução (obtido a partir do REG-DE pela Instrução anterior).
No caso do exemplo do item 11.2, se REG-DE contiver #8001 (Endereço da Linha Recuperada), a instrução da linha 1430 ficará "LD HL,(#8001)", com o que será carregado #800C no REG-HL (conteúdo dos BYTES #8001/#8002), que é o "Endereço de Início da Linha Seguinte à Linha Recuperada".

1440 Carrega no REG-A o conteúdo do REG-H.
1450 Compara o conteúdo do REG-A com o conteúdo do REG-H, na modalidade OR.
O indicador Z é ligado caso o BYTE resultante seja #00 e para que isto aconteça é necessário que ambos os registradores contenham #00.
Este teste, portanto, visa verificar se REG-HL contém #0000, o que indica a condição de "Fim de Programa".
1460 "Liga" o Indicador de Estado CY (SCF = SET CARRY FLAG - Liga Indicador de Transporte).
1470 Caso o Indicador Z tenha sido ligado na instrução 1450, a condição de "Fim de Programa" terá sido alcançada (dois primeiros BYTES da "Linha Recuperada" contém #0000).
Neste caso, é efetuado o RETORNO ao ponto de chamada a PXLIN, com as condições (C,Z) posicionadas.
1480 Adiciona uma unidade ao REG-DE.
1490 Adiciona uma unidade ao REG-DE.
Neste ponto o REG-DE passa a apontar para o terceiro BYTE da "Linha Recuperada" (o terceiro e quarto BYTES contém o "Número" desta linha).
1500 Carrega REG-DE no endereço "A06+2", que é o "Parâmetro de Endereço" da instrução seguinte (1510).
1510 Utilizando a mesma técnica descrita nas linhas 1420/1430, carrega no REG-DE o "Número da Linha Recuperada".
1520 Carrega no REG-HL o conteúdo do campo T03 do BIT-BASIC, onde está armazenado o "Número de Linha a Ser Comparado".
1530 Utiliza a rotina DCOMPR do BIOS para comparar REG-HL com REG-DE.
1540 Troca os conteúdos de REG-DE e REG-HL.
Com isto, REG-HL passa a conter o "Número da Linha Recuperada".
1550 Repete a operação da linha 1420
1560 Repete a operação da linha 1430.
Com isto, recolocamos no REG-DE o "Endereço de Início da "Linha Recuperada", que poderá ser utilizado pela rotina chamadora para repetir toda a seqüência de operações da rotina PXLIN, recuperando desta forma todas as linhas do programa, uma após a outra, até se alcançar a condição desejada.
1570 Caso a comparação efetuada na linha 1530 resulte como "Iguais" (Z), o que indica qual "Número da Linha Recuperada" é igual ao "Número Pesquisado", é efetuado o RETORNO sob a condição (NC,Z).
1580 "Complementa" o valor de CY.
1590 RETORNA ao ponto de chamada de PXLIN sob as condições (NZ,NC) ou (NZ,C), indicando que o "Número da Próxima Linha" é MAIOR ou MENOR do que o "Número Pesquisado".
OBSERVAÇÃO : As instruções das linhas 1540 a 1570 não alteram o posicionamento dos Indicadores CY e Z.

12 — ROTINA LISTPG

12.1 - OBJETIVO

O objetivo desta rotina é apresentar um programa BASIC na Tela do Monitor ou TV, a partir de uma linha determinada, até atingir a condição de "Tela Cheia" ou de "Fim do Programa", situação em que o processo é interrompido e o controle é devolvido ao Interpretador BASIC.
São novas funções de grande utilidade disponibilizadas pelo BIT-BASIC, especialmente quando se trata de programa com grande número de linhas, pois a listagem é feita "Sob Controle do Usuário".

12.2 - SINTAXE E FUNÇÕES DOS COMANDOS.

12.2.1 - LISTA PÁGINA A PARTIR DE UMA LINHA DETERMINADA.

Sintaxe : <<lll (HOTBIT) ou //lll (EXPERT)
Função : Lista página a partir da linha de número lll

Sintaxe : <[(HOTBIT) ou /[(EXPERT)
Função : Lista página a partir do início do programa

Sintaxe : << (HOTBIT) ou // (EXPERT)
Função : Avança uma linha na página atual

Exemplo: <<130 (HOTBIT) ou //130 (EXPERT)
Lista "Página" a partir da linha 130

12.2.2 - LISTA "PRÓXIMA PÁGINA"

Sintaxe : < (HOTBIT) ou / (EXPERT)
Função : Lista "Página" a partir da linha imediatamente seguinte à última linha apresentada na Tela anterior.

12.2.3 - FIXA "ÍNDICE DE PÁGINA"

Sintaxe : <<lll,i (HOTBIT) ou //lll,i (EXPERT)
lll = Número da linha
i = Índice ("0" a "9")
Função : Relaciona a linha de número lll ao Índice "i"

Exemplo : <<320,3 (HOTBIT) ou //320,3 (EXPERT)
O Índice de número "3" passa a "Apontar" para a linha de número 320

OBS. : Para ser utilizado em associação com o comando "Lista Página a partir de Índice", a seguir descrito. A diferença deste comando em relação ao do ítem 12.2.1 é a presença do segundo parâmetro (Índice).

12.2.4 - LISTA PÁGINA A PARTIR DE ÍNDICE

Sintaxe : <i (HOTBIT) ou /i (EXPERT)
i = Índice ("0" a "9")
Função : Lista página a partir da linha cujo número está associado ao índice "i"

Exemplo : <3 (HOTBIT) ou /3 (EXPERT)
Lista Página a partir da linha cujo número foi "Associado" ao Índice "3"
Equivale a <<320 ou //320 para o caso mostrado no exemplo do item 12.2.3
A diferença deste comando em relação ao do item 12.2.2, é a presença do índice após o "<" ou "/"

12.3 - CARACTERÍSTICAS

Esta rotina é acionada quando a primeira posição da linha de comando para o BIT-BASIC contém o caráter "<" (#3C - HOTBIT) ou "/" (#2F - EXPERT).

Antes de iniciar a apresentação a Tela é totalmente "Apagada" sendo as linhas "escritas" de cima para baixo. Isto permite uma maior velocidade em relação ao comando LIST do BASIC, em razão de não ser necessário "Deslocar para Cima" toda a Tela a cada vez que ela se encontra "Cheia" e nova linha deve ser inserida.

Se a última linha do programa não cabe na PÁGINA (Tela Completa) atual, ela é "Apagada" para não aparecer incompleta. Portanto, todas as linhas listadas pelo BIT-BASIC sempre aparecem "completas" na Tela.
É possível também "Identificar Rotinas" do programa BASIC com números de "0" a "9" (Índice), para posteriormente solicitar que este programa seja listado a partir destas rotinas.

Estas "Novas Funções" acrescentadas ao BASIC são extremamente úteis para se trabalhar com programas "Extensos", especialmente quando é necessário "Caminhar" neles.

A rotina apresenta também outros "Pontos de Entrada" ("Entry-Points") (LISTPG1, LISTPG2, FIMLST) que são utilizados por outras rotinas do BIT-BSIC que também necessitam apresentar linhas na tela.

A rotina LIST do Interpretador BASIC (que está colocada entre os endereços #522E e #5283 da ROM) não foi utilizada pelo BIT-BASIC porque sua codificação e pontos de entrada não atendem às necessidades deste.

12.4 - ROTINA ASSEMBLER (LISTAPG).

Este trecho da rotina LISTAPG determina qual a função a executar e efetua a preparação de sua execução. Executa também a função de de "Fixar Índice de Página".

CONDIÇÕES DE ENTRADA

REG-HL - Aponta para o segundo caráter da linha de comando do BIT-BASIC.

T03 - Contém o Número da Última Linha da "Página Anterior".

T01 - Contém o Número da Primeira linha da Página Anterior.

CONDIÇÕES DE SAÍDA

T01 - Número da Primeira Linha da página listada.

T03 - Número da Última Linha da página listada.

```
1620 LISTAPG:  CALL RST10
1630           JR   Z,B07
1640           JR   C,B05
1650           CP   #XX              (#3C  HOTBIT  -  #2F  EXPERT)
1660           JR   NZ,B04
1670           CALL PARM
1680           JR   C,B01
1690           JR   NZ,B02
1700 B01:      LD   DE,(T01)
1710           INC  DE
1720 B02:      LD   (T03),DE
1730           CALL RST10
1740           CP   #2C
1750           JR   NZ,B07
1760           CALL RST10
1770           JR   NC,B07
1780           CALL INDICE
1790           LD   BC,(T03)
1800           LD   (B03+2),HL
1810 B03:      LD   (#0000),BC
1820           JP   RET04
1830 B04:      CP   #5B
1840           JR   NZ,B07
1850           LD   DE,#0000
1860           LD   (T03),DE
1870           JR   B07
1880           DEFS 18
1890 B05:      CALL INDICE
1900           LD   (B06+1),HL
1910 B06:      LD   HL,(#0000)
1920           LD   (T03),HL
1930 B07:      LD   A,#0C
1940           RST  #18
```

1620 Obtém próximo caráter da linha de comando.

1630 Caso encontre a condição de "Fim de Linha" (Z) assume a função "Lista Próxima Página" (a linha de comando continha somente o caráter "<" ou "/").
Isto será feito a partir da instrução 1930, de nome B07.

1640 Caso encontre um caráter numérico (C) imediatamente após o caráter "<" ou "/", assume a função de "Lista Página a Partir de Índice", que está implementada a partir da Instrução 1890 (B05).

1650 Se as condições anteriores não se verificarem, testa se o caráter obtido é #3C ("<" - HOTBIT) ou #2F ("/" - EXPERT).

1660 Se não é "<" ou "/" desvia para a Instrução 1830 (B04).

**** Caso os dois primeiros caracteres da Linha de Comando para o BIT-BASIC sejam "<<" (HOTBIT) ou "//" (EXPERT), assume a função "Lista Página a Partir de Uma Linha Determinada" ou a função "Fixa Índice de Página".

1670 Efetua chamada à rotina PARM, cuja função é "Obter Parâmetro Numérico" (no REG-DE).

1680 Desvia para a instrução 1700 (B01) caso tenha sido obtido um parâmetro numérico inválido.

1690 Desvia para a instrução 1720 (B02) caso tenha sido obtido um parâmetro numérico válido e diferente de zero.

1700 Caso o parâmetro numérico seja zero ou inválido, coloca em REG-DE o número da "Primeira Linha da Página atual". (Assume função "Avança Uma Linha e Lista Página".)

1710 Soma 1 em REG-DE, para que seja pesquisada a linha "Seguinte à Primeira Linha da Página Anterior".

1720 Armazena o parâmetro numérico obtido a partir do comando BIT-BASIC (ou o número de linha recuperado pelas instruções 1700/1710) no "Campo de Trabalho" T03, (dois BYTES em #7B16/#7B17), onde é guardado o "Número da Linha Seguinte à Última Linha Apresentada na Tela" (o BIT-BASIC sempre lista uma nova página a partir do número contido neste campo).

1730 Obtém próximo caráter da linha de comando.
O objetivo é verificar se existe mais um parâmetro a ser recuperado (indicado pela presença de uma vírgula (#2C) após o primeiro parâmetro numérico).
Em caso negativo, assume a função "Lista Página a Partir de Uma Linha Determinada" (<< ou //).
Em caso afirmativo, assume a função "Fixa Índice de Página" (<<lll,i ou //lll,i).

1740 Compara caráter obtido com #2C (",").

1750 Se não é vírgula, desvia para a instrução 1930 (B07).

**** As linhas 1760 a 1820 implementam a função "Fixa Índice de Página".

1760 Obtém no REG-A o próximo caráter da linha de comando (Índice, "0" a "9").

1770 Se o caráter não é numérico, desvia para a instrução 1930 (B07 = Lista Página a partir do primeiro parâmetro obtido).

1780 Efetua chamada à rotina INDICE, encarregada de obter em REG-HL o endereço na tabela T02, onde deve ser guardado o parâmetro numérico anteriormente obtido, correspondente ao índice contido em REG-A.

1790 Carrega no REG-BC o primeiro parâmetro numérico obtido, que havia sido salvo no campo T03.

1800 Carrega o endereço contido no REG-HL (endereço pertencente à tabela T02), nos BYTES correspondentes ao operando de endereço da instrução seguinte (B03).

1810 Carrega na tabela T02 (posição correspondente ao endereço de REG-HL), o valor do REG-BC (número de linha do primeiro parâmetro).

1820 Encerra a função que "Fixa Índice de Página", retornando ao BASIC via rotina RET04 (retorno "Normal" ao BASIC).

**** As linhas 1830 a 1870 complementam a verificação da função solicitada, que pode ser ainda "<[" ou "<" (HOTBIT) ou "/[" ou "/" (EXPERT).

1830 Compara caráter obtido com "#5B" ("[").

1840 Se não é "[", desvia para a instrução 1930 (B07).
Com isto será executada a função "Lista Página" a partir dos valores já existentes em T01 e T03.

1850 Se caráter em REG-A é "[", carrega #0000 em REG-DE.

1860 Carrega #0000 em T03 para que a Página seja listada a partir da primeira linha do programa.

1870 Desvia para a instrução 1930 (B07).

1880 BYTES livres, para facilitar adaptações no BIT-BASIC.

**** As linhas 1890 a 1920 fazem a preparação para a função que "Lista Página a Partir de Índice", recuperando da tabela T02 o número da linha correspondente ao índice especificado.

1890 Efetua chamada à rotina INDICE, encarregada de obter em REG-HL o endereço na tabela T02 de onde deve ser recuperado o número de linha previamente armazenado, correspondente ao "Índice" contido em REG-A ("0" a "9").

1900 Carrega no operando de endereço da instrução B06 (linha 1910) o conteúdo de REG-HL, que contém o endereço desejado da tabela T02.

1910 Carrega em REG-HL os dois BYTES da tabela T02 que contém o número da linha desejada.
Se o número fornecido como índice for "0", REG-HL conterá o primeiro e segundo BYTES da tabela. Se o número for "1", conterá o terceiro e quarto BYTES, e assim por diante.

1920 Carrega no campo T03 o valor do REG-HL (Número da Linha Correspondente ao Índice).
Este número é "Salvo" para uso posterior.

**** As linhas 1930/1940 "Limpam a Tela" antes de iniciar a apresentação da linhas do programa.

1930 Carrega em REG-A o valor #0C.
1940 Efetua chamada à rotina OUTDO (OUTPUT DO - EXECUTA SAÍDA) do BIOS, cuja função é "Enviar um Dado ao Último Dispositivo Referenciado" (neste caso, a Tela).
Alguns caracteres têm finalidades especiais quando tratados por esta rotina (veja APÊNDICE 06).
Este é o caso do caráter #0C, cuja função é "Limpar a Tela e Voltar o Cursor para a Posição (1,1)", equivalente ao acionamento das teclas "CONTROL+L".
Em função deste procedimento, cada nova "Página" é apresentada pelo BIT-BASIC sempre "de cima para baixo".

12.5 - ROTINA ASSEMBLER (LISTPG1)

Este trecho da rotina LISTAPG percorre o programa BASIC até se posicionar na primeira linha a ser apresentada.

CONDIÇÕES DE ENTRADA :
T03 - Número da Linha a ser Procurada

CONDIÇÕES DE SAÍDA :
REG-DE - Endereço da Primeira Linha a Apresentar

```
1960 LISTPG1:  LD   DE,(#F676)
1970 B10:      CALL PXLIN
1980           JR   NC,LISTPG2
1990           JR   NZ,B10
```

1960 Carrega no REG-DE o conteúdo dos BYTES #F676/#F677 (Campo TXTTAB do BASIC = Endereço da primeira linha do programa).
A primeira linha a ser apresentada é procurada sempre desde o início do programa BASIC, em razão de que a cada nova inclusão/exclusão/alteração o Interpretador BASIC realoca todas as linhas do programa, com modificação de seus endereços na memória.
1970 Efetua chamada à rotina PXLIN, para obter a "Próxima Linha" do programa BASIC.
Esta rotina compara o número da linha obtida com o conteúdo do campo T03, e posiciona os Indicadores CY e Z para indicar as condições encontradas.

**** Neste ponto o campo T03 pode conter os valores:
1) "Zero", caso o comando atual seja "<<" ou "//" ("Listar Página a Partir do Início do Programa BASIC").
2) "111", caso o comando atual seja "<<111" ou "//111". ("Lista Página a Partir da Linha 111.)

3) "Número da Linha Seguinte à Última Linha Já Apresentada na Tela", caso o comando atual seja "<" ou "/" (este número terá sido salvo na passagem anterior pela rotina LISTAPG.
("Lista Próxima Página".)
4) "Número da Linha Correspondente ao Índice i", caso o comando atual seja "<i" ou "/i".
("Lista Página a Partir de Índice".)

1980 Caso o número da linha obtida seja maior ou igual a T03, desvia para a rotina LISTPG2, que iniciará a apresentação da Página na Tela.
1990 Caso o fim do programa não tenha sido atingido, desvia para a instrução 1970 (B10) para obter a próxima linha. Caso tenha sido alcançado o fim do programa segue para a Instrução seguinte (Rotina FIMLST).

15.6 - ROTINA ASSEMBLER (FIMLST)

Este trecho da rotina LISTAPG efetua os procedimentos para encerramento da listagem de uma Página na Tela.

```
2010 FIMLST:          XOR  A
2020                  LD   (T07),A
2030                  LD   HL,#0015
2040                  JP   RET02
```

2010 "Zera" o conteúdo de REG-A.
2020 Move zero (REG-A) para o campo T07.
O campo T07 é utilizado para indicar que devem ser apresentadas apenas as linhas que contém uma constante determinada (veja comando "=", ítem 19), e é sempre "Desligado" ao fim de cada Página apresentada.
2030 Carrega o valor #0015 no REG-HL.
2040 Desvia para a rotina RET02 que fará a passagem do caráter #15 à rotina de tratamento de Tela. Assim será executada a função especial "Apaga Toda a Linha Onde Está o Cursor" (veja APÊNDICE 06).
Procedendo desta forma, a última linha que seria apresentada na tela será sempre inteiramente "Apagada", ficando o seu número "Guardado" no campo T03 para que ela seja apresentada na "Próxima Página", caso solicitado.
Esta estratégia foi utilizada em razão de que as linhas de programa BASIC são de tamanho variável, e serve para garantir que nunca uma linha será "Parcialmente" mostrada na Tela.

12.7 - ROTINA ASSEMBLER (LISTPG2)

Este trecho da rotina LISTAPG apresenta na Tela as linhas de uma Página, uma a uma, até alcançar a condição de "Tela Completa" ou de "Fim de Programa".

CONDIÇÕES DE ENTRADA :
REG-HL - Número da Primeira linha
REG-BC - Endereço de Início da Primeira linha
REG-DE - Endereço de Início da Próxima Linha

(Para o exemplo de programa mostrado no ítem II-11.2, no caso da primeira linha, o conteúdo do REG-HL seria #000A (10), o do REG-BC #8001, e o do REG-DE seria #800C).

```
2070 LISTPG2:  LD   (T01),HL
2080 B11:      CALL #00B7
2090           JR   C,FIMLST
2100           PUSH DE
2110           CALL IMPLIN
2120           POP DE
2130           JR   C,FIMLST
2140           CALL PXLIN
2150           JR   NZ,B11
2160           JR   FIMLST
```

2070 "Salva" o "Número da Primeira Linha Apresentada na Tela" no campo T01.
2080 Efetua chamada à rotina BREAKX ("INTERRUPÇÃO-X") do BIOS, cuja função é "Verificar se Estão Pressionadas as Teclas CONTROL+STOP".
2090 Caso as teclas estejam sendo acionadas (neste caso o Indicador CY é "Ligado" pela rotina BREAKX), o programa é desviado para a rotina FIMLST, já descrita no ítem anterior.
Isto é feito para possibilitar ao usuário a interrupção da listagem antes do seu final.
2100 "Salva" na PILHA DO SISTEMA o REG-DE, que contém o "Endereço de Início da Próxima Linha" (REG-DE será utilizado/modificado pela rotina IMPLIN).
2110 Efetua chamada à rotina IMPLIN, que faz a apresentação de uma linha na Tela.
2120 Recupera o "Endereço de Início da Próxima Linha" da PILHA.
2130 Caso tenha sido atingida a condição de "Fim de Tela" (o Indicador CY é "Ligado" por IMPLIN neste caso), desvia para a rotina FIMLST, já descrita no ítem anterior.
2140 Obtém a "Próxima Linha" do programa BASIC.
2150 Caso não tenha sido alcançada a condição de "Fim de Programa" (indicador Z posicionado pela rotina PXLIN), desvia para a instrução 2080 (B11) desta rotina, para apresentação da nova linha na Tela.
2160 Caso tenha ocorrido o fim do programa antes de completada a Tela, desvia para a rotina FIMLST.

13 – ROTINA IMPLIN

13.1-OBJETIVO

O objetivo desta rotina é "Decodificar" a linha de programa BASIC, mostrar na Tela o "Número da Linha" e acionar a rotina IMPL que apresenta os demais caracteres na Tela.

13.2 - CONDIÇÕES DE ENTRADA
REG-HL - Número da Linha
REG-BC - Endereço de Início da Linha

13.3 - CONDIÇÕES DE SAÍDA
CY = C - Tela Completa
CY = NC - Tela ainda não atingiu a última linha inferior

13.4 - ROTINA ASSEMBLER

```
2190 IMPLIN:    LD   (T03),HL
2200            LD   HL,#0004
2210            ADD  HL,BC
2220            LD   BC,#5284
2230            CALL CALL
2240            LD   A,(T07)
2250            CP   #00
2260            JR   Z,B14
2270            CALL VERCHR
2280            CCF
2290            RET  NC
2300 B14:       LD   HL,(T03)
2310            LD   BC,#3412
2320            CALL CALL
2330            LD   A,#20
2340            RST  #18
2350            LD   HL,#F55D
2360            CALL IMPL
2370            RET
```

2190 "Salva" REG-HL (Número da Linha) no campo T03.
2200 Carrega #0004 em REG-HL.
2210 Soma ao REG-HL o REG-BC (Endereço de Início da Linha). Com isto, REG-HL passa a apontar para a "Posição Inicial da Linha +4", que é onde se inicia de fato o seu conteúdo, já que os dois primeiros BYTES contém o "Endereço da Próxima Linha" e os dois BYTES seguintes contém o "Número da Linha" (veja exemplo do ítem II-11.2, no qual REG-HL estaria apontando para #8005 no caso da linha 10, ou #8016 no caso da linha 30).

2220 Carrega no REG-BC o valor #5284.
2230 Aciona a rotina do BASIC encarregada de efetuar a "Decodificação" de uma linha de programa, "Desfazendo" a codificação interna efetuada pelo BASIC.
Esta rotina inicia no endereço #5284 da ROM, e por este motivo é necessário acessá-la por intermédio da rotina CALL.
Esta rotina sempre coloca no BUFFER a linha de programa BASIC "Desmontada" (a partir do endereço #F55E), na mesma forma com que ela foi originalmente digitada pelo usuário (exceto o "Número da Linha" que não é aí colocado).
2240 Carrega no REG-A o conteúdo do campo T07, que indica "Se É para Verificar ou Não" a presença de uma constante determinada na linha de programa (este campo é "Ligado" pela rotina VERCTE - VERIFICA CONSTANTE).
2250 Verifica se o campo T07 está "Desligado".
2260 Caso T07 esteja "Desligado", desvia para a instrução 2300 (B14), sem pesquisar a constante.
2270 Caso T07 esteja "Ligado", efetua chamada à rotina VERCHR (VERIFICA CARACTERES), cujo objetivo é checar se um determinado conjunto de caracteres, fornecidos por um comando "=" está presente na linha atualmente processada para decidir se ela deve ou não ser apresentada na Tela.
2280 "Inverte" o valor de CY.
A rotina VERCHR "Liga" CY se a constante não está presente, porém este Indicador é "Invertido" por esta instrução, para não retornar da rotina IMPLIN com o Indicador CY "Ligado" neste caso, o que indicaria a condição de "Tela Completa", quando não é esta a situação.
2290 RETORNA ao ponto de chamada da rotina IMPLIN, não listando a linha atual, caso a constante determinada não esteja presente nesta linha.
2300 Recupera em REG-HL o "Número da Linha Atual", salvo no campo T03 pela instrução 2190 no início desta rotina.
2310 Carrega no REG-BC o valor #3412.
2320 Aciona a rotina do BASIC encarregada de efetuar a apresentação de um "Número ASCII" na Tela, a partir do "Valor Binário" contido no REG-HL.
Esta rotina inicia no endereço #3412 da ROM, porém utiliza instruções de endereços ocupados pelo BIT-BASIC, sendo necessário acessá-la por intermédio da rotina CALL.
Como REG-HL contém o "Número da Linha", é este o número que será colocado na Tela.
2330 Carrega #20 no REG-A.
2340 Apresenta o caráter "Branco" ("#20") na Tela, em seguida ao "Número da Linha", utilizando a rotina OUTDO do BASIC.
2350 Carrega no REG-HL o valor #F55D, que é o "Endereço do BUFFER da Linha Decodificada, menos 1".
2360 Efetua chamada à rotina IMPL, a seguir descrita, que colocará o TEXTO DA LINHA na Tela.
2370 RETORNA ao ponto de chamada à rotina IMPLIN, para obtenção da "Próxima Linha" a ser apresentada.

14 - ROTINA IMPL

14.1 - OBJETIVO

O objetivo desta rotina é controlar a colocação de cada caráter da linha atual na Tela, a partir do BUFFER do BASIC que inicia em #F55E, até que toda a linha tenha sido apresentada ou até que tenha sido alcançada a "Última Linha Inferior da Tela".

14.2- CONDIÇÕES DE ENTRADA :

REG-HL - Endereço de Início do BUFFER, menos 1

14.3- ROTINA ASSEMBLER

```
2400 IMPL:      CALL RST10
2410            JR   Z,B16
2420            CALL IMPTELA
2430            RET  C
2440            JR   IMPL
2450 B16:       LD   A,#0A
2460            CALL IMPTELA
2470            RET  C
2480            LD   A,#01
2490            LD   (#F3DD),A
2500            RET
```

2400 Obtém próximo caráter da linha a ser listada.
2410 Caso tenha sido atingida a condição de "Fim de Linha" (indicada por #00), desvia para a instrução 2450 (B16).
2420 Efetua chamada à rotina IMPTELA, a seguir descrita, que apresenta o caráter na Tela e verifica se a última linha foi alcançada (Liga Indicador CY).
2430 RETORNA ao ponto de chamada de IMPL caso a última linha tenha sido atingida.
2440 Desvia para a instrução 2400 (IMPL) para obter e apresentar o próximo caráter.

**** As instruções das linhas 2450 a 2500 são executadas quando da condição de "Fim de Linha" e efetuam a movimentação do CURSOR para o início da próxima linha da Tela.

2450 Carrega no REG-A o valor #0A.
2460 Envia para a tela o caráter #0A, utilizando para isto a rotina IMPTELA.
Esta ação resulta na execução de um "Procedimento Especial" pela rotina de Administração da Tela (veja APÊNDICE 06), que neste caso é a função de "Mudança de Linha - Line Feed", acionada por aquele caráter.
O cursor estará posicionado na linha seguinte à última linha listada.

2470 Caso a última linha inferior da Tela tenha sido alcançada, RETORNA ao ponto de chamada a IMPL.
2480 Carrega #01 em REG-A.
2490 Armazena o valor #01 no campo de nome CSRX do BASIC (CURSOR POSITION X), localizado no BYTE #F3DD, onde é armazenado o valor da posição "X" atual do CURSOR (posição "dentro" de uma determinada linha, ou "número da coluna").
O efeito deste procedimento é fazer com que o CURSOR passe a estar na primeira posição (Coluna) da linha posicionada pela instrução de 2460.
OBS.: A posição de memória de endereço #F3DC guarda o campo CSRY (CURSOR POSITION Y) onde o BASIC armazena o valor da posição "Y" do cursor (Linha da Tela).
Podemos utilizar as variáveis CSRX e CSRY para movimentar o CURSOR pelas posições que desejarmos na Tela.
2500 RETORNA ao ponto de chamada da rotina IMPL.

15 - ROTINA IMPTELA

15.1- OBJETIVO

O objetivo desta rotina é apresentar na TELA do Monitor ou TV conectada ao seu MSX o caráter contido no REG-A, utilizando para isto a rotina do BIOS de "Administração de Tela" OUTDO (EXECUTA SAÍDA - RST #18).

15.2 - CONDIÇÕES DE ENTRADA

REG-A - Caráter a ser apresentado

15.3 - CONDIÇÕES DE SAÍDA

CY = C - Última linha da Tela alcançada
CY = NC - Última linha da Tela não alcançada

15.4 - ROTINA ASSEMBLER

```
2530 IMPTELA:  LD   C,A
2540           LD   A,(T18)
2550           CP   #00
2560           JR   NZ,B17
2570           LD   A,(#F3DC)
2580           CP   #17
2590           CCF
2600           RET  C
2610 B17:      LD   A,C
2620           RST  #18
2630           RET
```

2530 "Salva" o caráter de REG-A no REG-C.
2540 Carrega no REG-A o campo T18, que indica se está sendo executado o comando ".c" ou ".m" do BIT-BASIC (Copia ou Move linhas - Veja rotina COPMOV).
2550 Compara REG-A com #00.
2560 Se o campo T18 está "Ligado" (REG-A diferente de #00), o que indica que está sendo processado um comando de CÓPIA ou MOVIMENTO de linhas, desvia para a Instrução 2610 (B17), com o que o caráter é enviado para a tela sem verificar se a última linha inferior da Tela foi atingida (tais funções não podem ser interrompidas por esta condição).

**** Caso o comando atual não seja ".c" ou ".m", verifica se o caráter a ser apresentado vai ou não para a última linha da Tela, o que é feito pelas Instruções 2570 a 2600. Estas Instruções utilizam para esta verificação o próprio controle do BIOS sobre o posicionamento do CURSOR.

2570 Carrega no REG-A o conteúdo do campo CSRY (#F3DC - número da linha da Tela onde o CURSOR está localizado).
2580 Compara a "Posição Vertical" atual do CURSOR com 23 = #17 (o indicador CY é "Ligado" caso o cursor esteja em uma linha "Menor" do que 23).

2590 "Inverte" o valor do indicador CY.
Após esta instrução o indicador CY estará "Ligado" caso o CURSOR esteja localizado na linha 23 (#17) ou em linha anterior a esta (a linha 23 é utilizada como "Última Linha Inferior" da Tela).
2600 RETORNA ao ponto de chamada de IMPTELA caso a última linha tenha sido alcançada, com o Indicador CY "Ligado" (C =Fim de Tela alcançado), para que sejam tomadas as providências devidas pela rotina chamadora.

**** As linhas 2610 a 2630, finalmente, colocam na TELA o caráter desejado.

2610 "Retorna" ao REG-A o valor salvo em REG-C pela Instrução 2530 (caráter a ser apresentado na Tela).
2620 Efetua chamada à rotina OUTDO (EXECUTA SAÍDA) do BIOS, que coloca o caráter desejado na Tela, ou executa a "Função Especial" por ele determinada (veja APÊNDICE 06).
2630 RETORNA ao ponto de chamada de IMPTELA, com o caráter desejado já apresentado e com o indicador CY "Desligado" (NC - Fim de Tela não alcançado).

16 – ROTINA INDICE

16.1- OBJETIVO

O objetivo desta rotina é calcular o endereço de tabela T02 onde deve ser colocado, ou de onde deve ser retirado, um "Número de Linha" correspondente a um "Índice" fornecido em comando do tipo "<<lll,i" ou do tipo "<i" (EXPERT "//lll,i" ou "/i").

16.2 - CARACTERÍSTICAS

Como o índice é comportado por um único dígito não é utilizada a rotina PARM para sua obtenção.

16.3 - CONDIÇÕES DE ENTRADA

REG-A - Índice (Número de "0" a "9" - Formato ASCII #30 a #39).

16.4- CONDIÇÕES DE SAÍDA

REG-HL - Endereço da tabela T02, correspondente ao "Número de Linha" relacionado ao Índice.

16.5- ROTINA ASSEMBLER

```
2660 INDICE:   SUB  #30
2670           ADD  A,A
2680           LD   D,#00
2690           LD   E,A
2700           LD   HL,T02
2710           ADD  HL,DE
2720           RET
```

2660 Subtrai #30 de REG-A.
Com isto seu conteúdo "0 a 9" (ACII = #30 A #39) continua a ser "0 a 9", porém em "Valor Binário", ou seja, "00000000 = #00" a "00001001 = #09".

2670 "Duplica" o valor de REG-A, adicionando o seu valor a ele próprio (cada Número de Linha ocupa dois BYTES).

2680 "Zera" o conteúdo de REG-D.

2690 Carrega REG-A em REG-E.
Com isto o REG-DE passa a conter o valor do "Deslocamento" do número de linha em relação ao "Início da Tabela".
- Se Índice = "0", Deslocamento = 0 (#0000).
- Se Índice = "1", Deslocamento = 2 (#0002).
- E assim por diante.

2700 Carrega em REG-HL o endereço de "Início da Tabela T02" (#7B02).

2710 Soma o "Endereço de Início da Tabela T02" com o "Deslocamento na Tabela" (REG-DE).
Assim teremos em REG-HL o "Endereço na Tabela T02" onde está o "Número de Linha" correspondente ao Índice fornecido.

2720 RETORNA ao ponto de chamada de INDICE.

17 - ROTINA VOLTLIN

17.1 - OBJETIVO

O objetivo desta rotina é "Retroceder" um certo número de linhas em um Programa BASIC a partir da "Primeira Linha Apresentada na Tela" e, em seguida, desviar para a rotina encarregada de "Listar Página" (LISTPG1/LISTPG2).

17.2 - CARACTERÍSTICAS.

Esta rotina é acionada quando a primeira posição da linha de comando para o BIT-BASIC contém o caráter "[" (#5B).

Se nos procedimentos de retorno o início do programa for alcançado, será mostrada uma Página a partir da primeira linha do programa.

17.3 - SINTAXE E FUNÇÕES DOS COMANDOS.

17.3.1 - RETROCEDE TRÊS LINHAS E LISTA PÁGINA.

Sintaxe : [
Função : Retrocede três linhas e lista página

A escolha de "Três" linhas foi feita ,neste caso, em razão de que este é o maior número de linhas que podemos retroceder com garantia de que nenhuma delas será "Pulada", já que mais de três linhas de 256 caracteres ultrapassam os limites da tela (com largura normal, WIDTH = 38).

17.3.2 - RETROCEDE UMA LINHA E LISTA PÁGINA.

Sintaxe : [[
Função : Retrocede uma linha e lista página

17.3.3 - RETROCEDE "N" LINHAS E LISTA PÁGINA.

Sintaxe : [[nnn
Função : Retrocede nnn linhas e lista página
Exemplo : [[30
Retrocede 30 linhas e lista página

17.4 - CONDIÇÕES DE ENTRADA

Comando "[", conforme sintaxe já descrita

17.5 - CONDIÇÕES DE SAÍDA

Condições requeridas pelas rotinas LISTPG1 ou LISTPG2

17.6 - ROTINA ASSEMBLER

**** As instruções das linhas 2750 a 2840 colocam no campo T04 a. "Quantidade de Linhas a Retroceder".

```
2750 VOLTLIN:  LD   DE,#0003
2760           LD   (T04),DE
2770           CALL PARM
2780           JR   C,C02
2790           JR   NZ,C01
2800           CALL RST10
2810           CP   #5B
2820           JR   NZ,C02
2830           LD   DE,#0001
2840 C01:      LD   (T04),DE
```

2750 Carrega #0003 em REG-DE.
2760 Carrega #0003 em T04 (quantidade linhas a retroceder).
2770 Efetua chamada à rotina PARM ("Obtém Parâmetro Numérico").
2780 Caso tenha ocorrido erro na obtenção do parâmetro (CY Ligado), desvia para a instrução C02 assumindo 03 para a quantidade de linhas a retroceder (T04).
2790 Caso o número obtido seja diferente de Zero, desvia para a instrução 2840 (C01).
2800 Caso o número obtido seja Zero (não foi fornecido parâmetro numérico), obtém "Próximo Caráter".
2810 Compara próximo caráter com "[" (#5B).
2820 Se o próximo caráter não é "[" desvia para a instrução 2850 (C02). Neste caso T04 contém 03.
2830 Carrega em REG-DE o valor "1" (#0001), caso o próximo caráter seja "[" (trata-se de um comando "[[" - Retrocede uma linha).
2840 Carrega em T04 o valor contido em REG-DE (que pode ser #0001 ou o valor do parâmetro numérico obtido).

**** As linhas 2850 a 2950 posicionam o BIT-BASIC na "Primeira Linha" da "Página Anterior" apresentada na Tela.
Se ela não está mais presente no programa (pode já ter sido eliminada) a linha seguinte é posicionada.
É a partir desta linha que o retrocesso é efetuado.

```
2850 C02:      LD   DE,(#F676)
2860           LD   HL,(T01)
2870           LD   (T03),HL
2880           LD   A,#0C
2890           RST  #18
2900 C03:      CALL PXLIN
2910           JR   NC,C04
2920           JR   NZ,C03
2930           JP   RET04
2940 C04:      PUSH BC
2950           POP  DE
```

2850 Carrega no REG-DE os BYTES #F676/#F677 que contém o"Endereço de Início da Primeira Linha de Programa BASIC" (campo TXTTAB do Interpretador BASIC).

2860 Carrega no REG-HL o "Número da Primeira Linha da Página Anterior", colocada no campo T01 pela rotina LISTPG2, Instrução 2070.

2870 Coloca o conteúdo de T01 em T03 (via REG-HL) para "Procurar" esta linha no programa BASIC (Rotina PXLIN).

2880 Carrega #0C no REG-A.

2890 "Apaga" a Tela (veja item II-12.4, Instrução 1940).

2900 Efetua chamada à rotina PXLIN, para obter o Endereço e o Número da "Próxima Linha" do programa BASIC.
A pesquisa começa a partir da primeira linha do programa em função do valor colocado no REG-DE pela instrução da linha 2850.
Será obtida uma linha de número "Igual ou Maior" que T03.

2910 Caso a rotina PXLIN "Devolva" a condição "NC", o que significa que o "Número da Linha Obtida" é "Igual ou Maior" ao "Número da Primeira Linha Tela Anterior" (Campo T03), desvia para a Instrução 2940 (C04) que dará continuidade ao comando de "Retrocesso".

2920 Caso a rotina PXLIN "Devolva" as condições (C,NZ), o que significa que o "Número da Linha Obtida" é "Menor" do que o número procurado (T03), desvia para a Instrução 2900 (C03) para obter e comparar a "Próxima Linha".

2930 Caso a condição retornada por PXLIN seja (C,Z), indicando "Fim do Programa", efetua retorno ao BASIC em "Condição Normal", sem mostrar qualquer linha na Tela.

2940 Coloca REG-BC na PILHA (endereço de início da linha Recuperada).

2950 Copia REG-BC para REG-DE (via PILHA).

**** As instruções das linhas 2960 a 3010 verificam se foi alcançado o "Inicio do Programa" durante os procedimentos de retorno, providenciando a sua listagem desde este ponto em caso afirmativo.

```
2960 C05:      LD    HL,(#F676)
2970           RST   #20
2980           JR    NZ,C06
2990           LD    DE,#0000
3000           LD    (T03),DE
3010           JP    LISTPG1
```

2960 Carrega em REG-HL o conteúdo dos BYTES #F676/#F677 (TXTTAB - "Endereco de Inicio de Primeira Linha do Programa").

2970 Compara REG-HL (Início Primeira Linha BASIC) com REG-DE (Início Linha Atual), via rotina DCOMPR do BIOS.

2980 Se REG-HL "Diferente" de REG-DE, (o que significa que a Linha Atual não é a Primeira), desvia para a instrução 3030 (C06) para "Retroceder Mais Uma Linha".
Se início do programa alcançado, segue com a Instrução 2990.

2990 Carrega #0000 no REG-DE

3000 Carrega #0000 em T03 (primeira linha a ser listada).
3010 Desvia para a rotina LISTPG1, que fará a apresentação do Programa BASIC a partir da primeira linha (T03 = #0000).

**** As instruções das linhas 3030 a 3080 retrocedem, BYTE a BYTE, a partir do endereço posicionado pelos procedimentos já descritos, até encontrar a posição de "Início da Linha Anterior Menos 1", que é o "Último Caráter" da linha anterior a esta (sempre contém #00).
Faremos sempre referência ao exemplo do ítem II-11.2 que descreve o armazenamento de programas BASIC.
Se estivéssemos posicionados no "Início da Linha 30" (Endereço #8012) ao final destas instruções, estaríamos posicionados no "Início da linha 20 menos 1" (#800B), que corresponde também ao "Final da Linha 10" (contém #00).
No caso de atingirmos o início do programa, estaríamos posicionados no endereço #8000, que sempre contém #00.

```
3030 C06:        PUSH DE
3040 C07:        DEC  DE
3050 C08:        DEC  DE
3060             LD   A,(DE)
3070             CP   #00
3080             JR   NZ,C08
```

3030 "Salva" REG-DE na PILHA (endereço primeira posição da linha atual) (#8012).
3040 Subtrai 1 do REG-DE, que passa a apontar para a "Última Posição" da linha anterior (#8011) (esta posição contém #00 e deve ser "Pulada", o que é feito pela instrução seguinte que reduz REG-DE de mais uma unidade).
3050 Subtrai 1 de REG-DE (#8010).
3060 Carrega no REG-A o caráter apontado por REG-DE.
3070 Compara REG-A com #00.
3080 Se REG-A não contém #00, desvia para a Instrução 3050 (C08) para obter e comparar o caráter anterior a este.

**** As instruções das linhas 3090 a 3160 verificam se a posição que contém #00 realmente corresponde a um "Final de Linha", pois é possível que no meio de uma linha também tenha sido armazenado um caráter #00 pelo BASIC (por exemplo, no endereço #8015).
Isto é efetuado verificando-se se as duas posições seguintes ao #00 contém o "Endereço de Início da Linha Seguinte" (os dois primeiros BYTES de qualquer linha sempre contém esta informação).
Para verificar se o #00 contido no endereço #800B do exemplo corresponde a um final da linha, o BIT-BASIC verifica se o conteúdo dos BYTES #800C/#800D é #8012, que corresponde ao endereço inicial da linha 30.
Como esta condição é verdadeira, conclui-se que o caráter #00 pesquisado está no final da linha anterior à linha BASIC de número 20 que, por sua vez, é a linha anterior à linha 30.
Com isto "Retrocedemos" uma linha no programa BASIC.

```
3090           INC  DE
3100           LD   (C09+1),DE
3110 C09:      LD   HL,(#0000)
3120           POP  BC
3130           AND  A
3140           SBC  HL,BC
3150           PUSH BC
3160           JR   NZ,C07
```

3090 Adiciona 1 ao REG-DE, que passa a apontar para o caráter seguinte ao que contém #00 (#800C).

3100 Carrega REG-DE no "Campo de Endereço" da Instrução 3110 que, para o caso do exemplo, passa a ser "LD HL,(#800C)".

3110 Carrega no REG-HL o conteúdo do endereço definido pela instrução anterior.
Após a sua execução, REG-HL conterá #8012, que é o valor armazenado nos BYTES #800C/#800D.

3120 Carrega o último valor da PILHA no REG-BC, que é o "Endereço de Início da Linha Anterior", aí colocado pela instrução 3030 (C06).

3130 "Zera" o Indicador de Estado CY.
Isto é feito para que o resultado da instrução seguinte não seja comprometido se CY estiver "Ligado".

3140 Subtrai de REG-HL o valor de REG-BC.
Esta instrução subtrai ainda o valor de CY deste resultado (nada será subtraído pois CY já foi "Zerado" na instrução anterior).
Esta instrução é utilizada para "Compararmos" os valores contidos em REG-HL e REG-BC.

3150 "Salva" novamente na PILHA o conteúdo de REG-BC, dali retirado pela instrução da linha 3120 (não altera o posicionamento dos Indicadores de Estado CY e Z).

3160 Se a subtração da linha 3140 resulta "NZ", o que indica que o conteúdo de REG-HL não era igual ao conteúdo de REG-BC (os dois BYTES após #00 não contém o endereço de início da próxima linha), desvia para a instrução 3040 (C07) para a continuação dos procedimentos de retrocesso até o próximo #00.
Se REG-HL igual a REG-BC (uma linha foi retrocedida), segue para o trecho a seguir descrito.

**** As instruções das linhas 3170 a 3240 verificam se já foi retrocedida a quantidade desejada de linhas (armazenada no campo T04 pelas instruções iniciais da rotina VOLTLIN).

```
3170           POP  BC
3180           LD   BC,(T04)
3190           DEC  BC
3200           LD   A,C
3210           OR   B
3220           JR   Z,C10
3230           LD   (T04),BC
3240           JR   C05
```

3170 Retira da PILHA o valor aí colocado pela Instrução 3150, com o objetivo de não deixar este valor "Pendente" na PILHA (ela deve sempre retornar à sua posição inicial).
3180 Carrega no REG-BC a quantidade de linhas que falta ainda retroceder (Campo T04).
3190 Subtrai 1 de REG-BC.
3200 Carrega REG-C em REG-A.
3210 Compara REG-B com REG-A na modalidade "OR".
O objetivo é verificar se REG-BC contém #00 (somente neste caso o Indicador Z estará "Desligado" após a execução desta instrução).
3220 Caso REG-BC contenha #00, é porque já retrocedemos à quantidade de linhas desejada e, então, é efetuado desvio para a instrução 3270 (C10), que encaminhará o programa para listar Página a partir da linha atual.
3230 Caso ainda não tenhamos retrocedido a quantidade de linhas desejada, armazena no campo T04 o conteúdo do REG-BC (quantidade de linhas a retroceder, menos 1).
3240 Desvia para a instrução 2960 (C05) para verificar se o início do programa BASIC foi atingido, continuando os procedimentos de retrocesso de linhas.

**** As linhas 3270 a 3350 efetuam a preparação para listar uma "Página" do programa BASIC, de acordo com os parâmetros exigidos pela rotina LISTPG2, a partir da linha obtida.

```
3270 C10:       PUSH DE
3280            INC  DE
3290            INC  DE
3300            LD   (C11+1),DE
3310 C11:       LD   HL,(#0000)
3320            POP  BC
3330            LD   (C12+2),BC
3340 C12:       LD   DE,(#0000)
3350            JP   LISTPG2
```

3270 Salva REG-DE na PILHA (endereço de início linha desejada).
3280 Adiciona 1 ao REG-DE.
3290 Adiciona 1 ao REG-DE. (passa a apontar para o Terceiro/Quarto BYTES da primeira linha a listar, #800E/#800F no caso do exemplo, que contém o "Número da Linha" = #0014 = 20).
3300 Carrega REG-DE no campo de endereço da instrução C11, que passa a ser "LD HL,(#800E)" para o caso do exemplo.
3310 Carrega no REG-HL o "Número da Linha" desejada (#0014 = 20 no caso do exemplo).
3320 Carrega PILHA no REG-BC (endereço da linha desejada).
3330 Carrega REG-BC no campo de endereço da instrução C12, que fica "LD DE,(#800C)" para o caso do exemplo.
3340 Carrega no REG-DE o "Endereço de Início da Próxima Linha" (#8012 para o caso do exemplo).
3350 Desvia para a rotina LISTPG2 que fará a apresentação de uma "Página" na Tela a partir da linha atual (linha 20, no caso do exemplo).

18 — ROTINA NOVOBAS

18.1 - OBJETIVO

O objetivo desta rotina é reposicionar os parâmetros internos do BIOS/BASIC de maneira a possibilitar o carregamento de um novo programa BASIC, após o programa atual, sem destruir o seu conteúdo.
Efetua ainda o "Retorno" ao primeiro programa, ou a "União" dos dois programas presentes na memória.

Estas novas funções disponíveis são muito úteis, por exemplo, quando queremos "inserir" no programa atual rotinas previamente preparadas e salvas.
O comando "Merge" do Interpretador BASIC apresenta, neste caso, as desvantagens de "Misturar" as linhas e de exigir o salvamento prévio das rotinas no formato ASCII (Save"..:..",A).

Outra utilidade das novas funções é "Olharmos" um programa armazenado em fita ou disco sem destruirmos o programa atual e, opcionalmente, a ele "Retornarmos" sem necessidade de NOVA operação de "Carregamento".

18.2 - SINTAXE E FUNÇÕES DOS COMANDOS

18.2.1 - REPOSICIONA VARIÁVEIS

Sintaxe : .z
Função : Variáveis BASIC passam a apontar para nova "Posição Inicial" de carga de programas (após o final do Programa Atual = Programa Pimário).
A carga do Programa Secundário poderá ser feita com as funções normais de "LOAD" do BASIC.

18.2.2 - RETORNA AO PROGRAMA PRIMÁRIO

Sintaxe : .zr
Função : Retorna ao programa BASIC "Primário".
(O programa "Secundário" é perdido.)

18.2.3 - UNIÃO DOS PROGRAMAS

Sintaxe : .zu
Função : Os programas Primário e Secundário se fundem num só.

18.3 - CARACTERÍSTICAS

Esta rotina é acionada quando as primeiras posições do comando BIT-BASIC contém os caracteres ".z".
O caráter seguinte define a opção desejada (Posicionamento, Retorno ou União).
Na função de "Preparação para Carga" é reservada área "Após" o programa corrente, que permanece na memória, porém "Invisível".
Se não há nenhum programa carregado não é aceita a "Preparação para Carga" de programa adicional, apresentando a mensagem de "ERRO DE SINTAXE".

É possível carregar apenas um programa "Adicional". Portanto, se a solicitação de carregamento for repetida por duas vezes consecutivas, será apresentada a mensagem "ERRO DE SINTAXE". A operação de "União" não "Mistura" as linhas dos programas. O novo programa permanece "Após o Primeiro" com a numeração original, mesmo que a numeração das linhas seja inferior à do primeiro programa (as linhas podem ser renumeradas posteriormente à operação de União).
Com a utilização das novas funções de "Cópia" e "Movimento" do BIT-BASIC poderemos, então, colocar a nova rotina no ponto do programa em que desejarmos.

18.4 - ROTINA ASSEMBLER (NOVOBAS).

Esta rotina determina a função a ser executada e efetua o "Reposicionamento das Variáveis".

**** As instruções das linhas 3380 a 3420 determinam a função a ser executada.

```
3380 NOVOBAS:  CALL RST10
3390           CP   #72
3400           JP   Z,RETBAS
3410           CP   #75
3420           JP   Z,UNIBAS
```

3380 Obtém, em REG-A, o caráter seguinte ao caráter "z".
3390 Compara REG-A com #72 (Caráter "r").
3400 Se REG-A = "r", desvia para a rotina RETBAS (Retorno ao programa Primário).
3410 Compara REG-A com #75 (Caráter "u").
3420 Se REG-A = "u", desvia para a rotina UNIBAS (União dos programas Primário e Secundário).

**** As instruções das linhas 3430 a 3460 não permitem que mais de um programa adicional seja carregado.

```
3430 D05:      LD   BC,(T06)
3440           LD   A,B
3450           OR   C
3460           JP   NZ,RET03
```

3430 Carrega T06 em REG-BC.
Este campo contém #0000 quando não há programa adicional carregado, e um valor diferente deste após a carga de um programa secundário.
(Retorna a #0000 após operação de "Retorno" ou "União".)
3440 Carrega REG-B em REG-A.
3450 Compara REG-A com REG-C na modalidade "OR".
3460 Se REG-BC (T06) não contém #0000, desvia para RET03. Como o comando original (.z) ainda está no BUFFER (#F55E), e como a rotina RET03 desvia para o endereço #4173 no qual o BASIC processa normalmente o conteúdo deste BUFFER, o resultado será a apresentação da mensagem "ERRO DE SINTAXE", já que ".z" é inválido para o BASIC.

**** As linhas 3470 a 3520 verificam se já há um programa BASIC carregado.

```
3470            LD   BC,(#F676)
3480            LD   (D06+2),BC
3490 D06:       LD   BC,(#0000)
3500            LD   A,B
3510            OR   C
3520            JP   Z,RET03
```

3470 Carrega no REG-BC o endereço de "Início da Primeira Linha do Programa" (campo TXTTAB do BASIC).
3480 Carrega REG-BC no campo de endereço da instrução 3490 (D06).
3490 Carrega no REG-BC o conteúdo dos dois primeiros BYTES do programa.
3500 Carrega REG-B em REG-A.
3510 Compara REG-A com REG-C na modalidade OR.
3520 Caso o resultado da comparação anterior seja "Zero" (Indicador Z "Ligado") é porque REG-BC contém #0000, o que indica a condição de "Fim de Programa" já na primeira linha, ou seja, não há qualquer programa carregado.
Nesta situação o BIT-BASIC não aceita o carregamento de um "Programa Adicional" e é efetuado desvio para a rotina RET03, o que resultará na apresentação da mensagem "ERRO DE SINTAXE".

**** As linhas 3530 a 3570 "procuram" o final do programa BASIC corrente para posicionar neste ponto as variáveis BASIC.

```
3530            LD   DE,(#F676)
3540            LD   (T06),DE
3550 D07:       CALL PXLIN
3560            JR   NC,D07
3570            JR   NZ,D07
```

3530 Carrega no REG-DE o endereço de "Início do Programa Atual" (campo TXTTAB do BASIC).
3540 "Salva" TXTTAB em T06 para possibilitar o "Retorno" posterior ao programa atual. Este campo é também utilizado para indicar se há ou não "Programa Secundário" carregado.
3550 Efetua chamada à rotina PXLIN, para obter a "Próxima Linha" do programa BASIC.
3560 Desvia para a instrução D07 para obter a "Próxima Linha" BASIC caso a condição de "Fim de Programa" não tenha sido atingida (C,Z).
3570 Idem instrução 3560.
Com o procedimento das linhas 3560 e 3570, a rotina PXLIN será repetida até "Devolver" a condição de "Fim de Programa", quando o controle passará então para a instrução seguinte (Instrução 3580).

**** As instruções das linhas 3580 a 3600 efetuam o reposicionamento das variáveis que apontam para o "Início do Programa BASIC".
Estas variáveis passarão a apontar para um endereço de memória RAM "Após" o programa Primário.
A inclusão posterior de um programa BASIC se dará a partir deste ponto, preservando intacto na memória o programa atualmente carregado.
São as seguintes as variáveis BASIC/BIOS envolvidas no problema em questão (veja APÊNDICE 01):
(Cada uma delas ocupa dois BYTES na memória RAM, e todas são automaticamente "Inicializadas" pelo BIOS no momento em que o microcomputador é "Ligado".)

```
TXTTAB (#F676) - TEXT TABLE (Tabela de Texto)
                 Início do "Texto BASIC" (Programa BASIC)
BOTTOM (#FC48) - BOTTOM (Fundo)
                 Início da memória RAM utilizada p/sistema
VARTAB (#F6C2) - VARIABLE TABLE (Tabela de Variáveis)
                 Início "Área de Variáveis Simples"
ARYTAB (#F6C4) - ARRAY TABLE (Tabela de Arranjos)
                 Início "Tabela de Arranjos do BASIC"
STREND (#F6C6) - STORAGE END  (Fim de Memória)
                 Fim da memória "Em Uso"
```

Destas variáveis somente as duas primeiras (TXTTAB e BOTTOM) serão diretamente modificadas pelo BIT-BASIC.
As demais serão atualizadas pelo próprio BIOS, por um comando CLEAR acionado a partir do BIT-BASIC.
Na situação inicial, quando o micro é ligado, BOTTOM aponta para #8000, que é a primeira posição da RAM, e TXTTAB aponta para #8001, primeira posição do programa.

```
3580          LD   (#F676),BC
3590          DEC  BC
3600          LD   (#FC48),BC
```

3580 Carrega REG-BC em TXTTAB.
REG-BC contém o endereço do "Fim do Programa".
Portanto, estamos fazendo com que a variável TXTTAB, que indica o "Início do Texto/Programa", passe a apontar para o BYTE imediatamente seguinte à última linha do programa BASIC atual.
Como estas posições já contém #0000, já está presente a condição de "Nenhum Programa Carregado" (se a listagem do novo programa for solicitada, nada aparecerá).

3590 Subtrai 1 de REG-BC, que passa portanto a apontar para o último BYTE da última linha do programa BASIC atual (este BYTE sempre contém #00).

3600 Carrega REG-BC em BOTTOM.
Portanto, estamos fazendo com que a variável BOTTOM, que indica o "Início da RAM", passe a apontar para o último BYTE da última linha do programa BASIC atual.

**** As instruções das linhas 3610 a 3650 providenciam o retorno ao BASIC com a execução da instrução CLEAR, a qual inicializa os demais parâmetros já descritos.

```
3610 DOB:       LD   HL,T05
3620            LD   DE,#F55E
3630            LD   BC,#06
3640            LDIR
3650            JP   RET03
```

3610 Carrega em REG-HL o endereço do campo T05, onde está armazenada a constante "CLEAR".
3620 Carrega #F55E em REG-DE.
3630 Carrega #0006 em REG-BC.
3640 Movimenta ("Copia") o número de BYTES indicado pelo REG-BC (06 BYTES) a partir do endereço apontado por REG-HL (T05), para o endereço apontado por REG-DE (#F55E). Com isto colocamos a constante "CLEAR" a partir do endereço #F55E (BUFFER BASIC), seguida de #00, pronta para ser executada.
3650 Desvia para a rotina RET03, que fará com que o BASIC execute comandos armazenados a partir de #F55E como se tivessem sido entrados via teclado (sem apresentá-los na Tela).
A partir deste momento podemos então carregar um novo programa BASIC sem destruir o conteúdo daquele já carregado.

18.5 - ROTINA ASSEMBLER (RETBAS)

Esta rotina retorna as variáveis BASIC à situação original, de forma a termos novamente acesso ao programa que ficou "Escondido" pela rotina NOVOBAS.
Providencia também a execução da instrução CLEAR pelo BIOS.

**** As instruções das linhas 3680 a 3730 confirmam se há programa secundário carregado e posicionam o campo T06.

```
3680 RETBAS:    LD   BC,(T06)
3690            LD   A,B
3700            OR   C
3710            JP   Z,RET03
3720            LD   HL,#0000
3730            LD   (T06),HL
```

3680 Carrega T06 (campo ende é "Salvo" o endereço de inicio do programa primário) em REG-BC.
3690 Carrega REG-B em REG-A.
3700 Compara REG-A com REG-C na modalidade OR.
3710 Se T06 (REG-BC) contém #0000, o que indica que "Não há Programa Secundário Carregado", não aceita o comando de "Retorno", apresentando a mensagem "ERRO DE SINTAXE" via rotina RET03.

3720 Carrega #0000 em REG-HL.
3730 "Desliga" o campo T06, aí colocando #0000, para registrar o fato de que não haverá programa secundário carregado (após a conclusão da rotina RETBAS).

**** As instruções das linhas 3740 a 3810 "Retornam" as variáveis BASIC aos seus valores originais e encaminham a execução da instrução CLEAR.

```
3740          LD   HL,(#F676)
3750          LD   (#F676),BC
3760          DEC  BC
3770          LD   (#FC48),BC
3780          LD   (HL),#00
3790          INC  HL
3800          LD   (HL),#00
3810          JP   D08
```

3740 Carrega em REG-HL o "Endereço de Inicio" do programa secundário (aí colocado pela rotina NOVOBAS).
3750 Carrega no campo TXTTAB o conteúdo do campo T06 (REG-BC - "Endereço de Início" do programa primário, aí colocado pela Instrução 3680).
3760 Subtrai 1 de REG-BC.
3770 Coloca o valor de REG-BC no campo BOTTOM.
3780 Coloca #00 no endereço apontado por REG-HL ("Primeiro BYTE imediatamente após o primeiro programa").
3790 Incrementa REG-HL de uma unidade.
3800 Coloca #00 no "Segundo BYTE após o programa primário". As instruçoes 3780 e 3800 restauram o conteúdo #0000 nestes dois BYTES para indicar a condição de "Fim de Programa".
3810 Desvia para a instrução 3610(D08) que encaminhará a execução da instrução CLEAR pelo BIOS.

18.6 - ROTINA ASSEMBLER (UNIBAS)

Esta rotina efetua a "União" dos programas BASIC "Primário" e "Secundário" presentes na memória RAM.
Isto é feito "Retornando ao Valor Original" as variáveis BASIC, sem no entanto restaurar a condição de "Fim do Programa Primário", com o que os dois programas presentes na memória ficam "Emendados", um imediatamente após o outro.
(É importante ressaltar que a numeração das linhas não é alterada.)
A rotina encaminha também a execução da instrução CLEAR pelo BIOS.

```
3840 UNIBAS:    LD    BC,(T06)
3850            LD    A,B
3860            OR    C
3870            JP    Z,RET03
3880            LD    HL,#0000
3890            LD    (T06),HL
3900            LD    (#F676),BC
3910            DEC   BC
3920            LD    (#FC48),BC
3930            JP    DOB
```

3840 Carrega T06 em REG-BC (Endereço de Início do Programa Secundário).
3850 Carrega REG-A em REG-B.
3860 Compara REG-A com REG-C na modalidade OR.
3870 Se REG-BC = #0000, o que indica que não há programa secundário carregado, não aceita comando de "União", apresentando mensagem "ERRO DE SINTAXE" via rotina RET03.
3880 Carrega #0000 em REG-HL.
3890 "Desliga" o campo T06, aí colocando #0000, para registrar o fato de que não haverá programa secundário carregado (após a conclusão da rotina UNIBAS).
3900 Retorna ao campo TXTTAB o seu valor original (REG-BC = T06).
3910 Subtrai 1 de REG-BC.
3920 Armazena REG-BC no campo BOTTOM.
Com isto as variáveis TXTTAB e BOTTOM retornam aos valores que correspondem ao primeiro programa BASIC (#8001 e #8000 em condições normais).
O "Fim do Programa Primário" não é restaurado, permanecendo o endereço de "Fim do Programa Secundário".
3930 Desvia para a Instrução 3610 (DOB) que encaminhará o retorno ao BASIC, com execução da instrução CLEAR.

19 — ROTINA VERCTE

19.1 - OBJETIVO

] objetivo desta rotina é procurar em um programa BASIC as linhas que contém uma dada constante, apresentando na Tela estas linhas "Prontas para Edição".

Esta nova função disponibilizada pelo BIT-BASIC é útil, por exemplo, quando precisamos modificar o nome de uma variável em vários pontos de um programa BASIC, ou quando precisamos verificar as linhas onde está sendo utilizado um determinado comando ou variável.
São apresentadas na Tela somente as linhas do programa onde esta constante é referenciada, prontas para serem normalmente "Editadas" (Modificadas, Excluídas) pelas funções disponíveis no BASIC e no BIT-BASIC.
Sem estas funções teríamos que percorrer "Visualmente" todo o programa, selecionando as linhas desejadas.

19.2 - SINTAXE E FUNÇÕES DOS COMANDOS.

19.2.1 - PESQUISA CONSTANTE

Sintaxe : = ccc
Função : Pesquisa as linhas do programa BASIC e lista na Tela aquelas que contém a constante ccc (até 20 posições).
Exemplos: =X1% Procura linhas que contém "x1%"
=PRINT Procura linhas que contém "PRINT"

19.2.2 - CONTINUAÇÃO DE PESQUISA.

Sintaxe : ==
Função : Continuação da pesquisa em caso de "Tela Completa".

19.3 - CARACTERÍSTICAS.

Esta rotina é acionada quando a primeira posição do comando BIT-BASIC contém o caráter "=" (#3D).
As linhas do programa BASIC que contém a constante desejada ficam disponíveis para edição sob o BASIC/BIT-BASIC.
A pesquisa é sempre iniciada pela primeira linha do programa e os caracteres alfabéticos são pesquisados tanto na condição de "Minúculos" como de "Maiúsculos".
Se as linhas que contém a constante desejada ocupam toda a Tela a listagem é interrompida. É possível "Continuar" a pesquisa utilizando o comando "=="

Inicialmente, a constante é obtida e "Salva", em uma área de trabalho, para em seguida ser efetuado o desvio para as rotinas encarregadas de Listar Página (LISTAPG) que, por sua vez, se utilizam de outra rotina (VERCHR) para verificar em cada uma das linhas se a constante está ou não presente.

19.4 - ROTINA ASSEMBLER (VERCTE)

**** Esta rotina "Salva" a constante a ser pesquisada e desvia para a rotina que "Lista Página".
As instruções 3960 a 4010 verificam a função a ser executada (Pesquisa ou Continuação).

```
3960 VERCTE:   CALL RST10
3970           JP   Z,RET03
3980           CP   #3D
3990           JR   Z,E03
4000           LD   BC,#0000
4010           LD   (T03),BC
```

3960 Obtém próximo caráter do comando BIT-BASIC.
3970 Se não há nenhum caráter, desvia para RET03 (ERRO).
3980 Compara REG-A com "=" (#3D).
3990 Se próximo caráter é "=" desvia para 4150 (E03) que fará a "Continuação" da pesquisa (Comando "==").
4000 Carrega #0000 em REG-BC.
4010 Carrega #0000 em T03 (número da próxima linha a ser listada), com o objetivo de "Forçar" a pesquisa desde a primeira linha do programa.

**** As instruções das linhas 4020 a 4140 obtém e salvam a constante a ser pesquisada.

```
4020           LD   DE,T09
4030           LD   (DE),A
4040           INC  DE
4050 E01:      CALL RST10
4060           JR   Z,E02
4070           LD   (DE),A
4080           INC  DE
4090           LD   A,(DE)
4100           CP   #FF
4110           JP   Z,RET03
4120           JR   E01
4130 E02:      XOR  A
4140           LD   (DE),A
```

4020 Carrega em REG-DE o endereço do CAMPO T09 (onde a constante a ser pesquisada é armazenada).
4030 Carrega REG-A (primeiro caráter da constante) no primeiro BYTE do campo T09 (apontado pelo REG-DE).
4040 Adiciona 1 ao REG-DE, que passa a apontar para o BYTE seguinte de T09.
4050 Obtém próximo caráter da constante.
4060 Caso tenham terminado os caracteres fornecidos, desvia para a instrução 4130 (E02).
4070 Carrega REG-A (próximo caráter) em T09.
4080 Adiciona 1 ao REG-DE.

4090 Carrega no REG-A a posição de T09 "Seguinte ao Último Caráter Armazenado".
O objetivo é verificar se já foi atingido o final do campo de trabalho T09 (20 BYTES).
4100 Compara REG-A com #FF (o final de T09 é indicado por #FF.)
4110 Caso o final de T09 tenha sido alcançado, desvia para a rotina RET03 (ERRO).
4120 Desvia para a Instrução 4050 (E01) para obter e salvar o próximo caráter da constante.
4130 "Zera" o conteúdo de REG-A (após toda a constante ter sido alocada em T09).
4140 Coloca #00 logo após a constante armazenada (para funcionar como indicador de "Fim da Constante").

**** As instruções das linhas 4150 a 4210 efetuam os preparativos finais para o início da pesquisa e desviam para a rotina que "Lista Página".

```
4150 E03:       LD   A,#FF
4160            LD   (T07),A
4170            LD   A,#0C
4180            RST  #18
4190            LD   HL,T08-1
4200            CALL IMPL
4210            JP   LISTPG1
```

4150 Carrega #FF em REG-A.
4160 "Liga" com #FF o campo T07, para indicar à rotina de "Lista Página" que está sendo processado um comando de "Pesquisa de Constante".
4170 Carrega #0C em REG-A.
4180 "Limpa" a Tela (veja Instrução 1940, ítem II-12.4).
4190 Carrega em REG-HL o "Endereço da Constante Armazenada -1" (precedida do caráter "=").
4200 Efetua a apresentação na tela da constante fornecida, precedida de "=".
4210 Desvia para o Ponto de Entrada LISTPG1 da rotina LISTAPG a qual fará a apresentação de uma Página somente com linhas que contenham a constante solicitada. (Esta verificação será feita pela rotina VERCHR, que será chamada em razão do campo T07 estar "Ligado".)

19.6 - ROTINA ASSEMBLER (VERCHR)

**** Esta rotina é "Chamada" pela rotina que "Lista Página" (IMPLIN), para cada linha do programa BASIC, verificando se a constante fornecida no comando "=" do BIT-BASIC está ou não presente.

```
4240 VERCHR:    LD   HL,#F55D
4250 E10:       LD   DE,T08
4260            PUSH HL
4270 E11:       INC  HL
4280            INC  DE
4290            LD   A,(DE)
4300            CP   #00
4310            JR   Z,E13
4320            LD   A,C
4330            LD   A,(HL)
4340            CP   #00
4350            SCF
4360            JR   Z,E13
4370            CP   C
4380            JR   Z,E11
4390            CP   #41
4400            JR   C,E12
4410            CP   #5B
4420            JR   NC,E12
4430            ADD  A,#20
4440            CP   C
4450            JR   Z,E11
4460 E12:       POP  HL
4470            INC  HL
4480            JR   E10
4490 E13:       POP  HL
4500            RET
```

4240 Carrega #F55D em REG-HL (endereço onde a linha BASIC a ser pesquisada é colocada, já "Decodificada").
4250 Carrega em REG-DE o endereço do campo T08 (constante a ser pesquisada precedida de "=").
4260 Salva REG-HL na PILHA.
4270 Soma 1 em REG-HL.
4280 Soma 1 em REG-DE.
4290 Carrega em REG-A o próximo caráter da constante.
4300 Compara REG-A com #00.
4310 Se REG-A = #00, o que significa que toda a constante já foi pesquisada, desvia para a instrução 4490 (E13), que fará o retorno ao ponto de chamada de VERCHR com a condição "NC" no Indicador CY, indicando que a presente linha "Contém" a constante fornecida.
4320 "Salva" REG-C em REG-A.
4330 Carrega em REG-A o próximo caráter da Linha BASIC.
4340 Compara REG-A com #00.
4350 "Liga" o Indicador de Estado CY.
4360 Se REG-A = #00, que significa que o "Fim da Linha BASIC" foi alcançado, desvia para a instrução 4490 (E13), fazendo o retorno ao ponto de chamada de VERCHR com a condição "C" no Indicador CY, indicando que a presente linha "Não Contém" a constante fornecida.

**** As instruções das linhas 4370 a 4450 comparam o caráter da constante fornecida com o caráter da linha BASIC, nos formatos "Maiúsculo" e "Minúsculo".
(Todos os caracteres da constante já foram convertidos pára "Minúsculos" na rotina INICIO.)

4370 Compara REG-A com REG-C (Caráter "Minúsculo" da Constante com Caráter da Linha BASIC).
4380 Se REG-A = REG-C (caracteres iguais), desvia para a instrução 4270 (E11) para comparar os próximos caracteres.
4390 Compara REG-A com "A" (#41).
4400 Se REG-A menor do que #41, o que indica "Caráter da Constante Não-Alfabético", desvia para 4460 (E12).
4410 Compara REG-A com #5B (caráter imediatamente seguinte a "Z" = #5A).
4420 Se REG-A não é menor do que #5B (caráter não-alfabético), desvia para 4460 (E12).
4430 "Transforma" caráter alfabético "Maiúsculo" de REG-A em "Minúsculo".
4440 Compara REG-A com REG-C.
4450 Se REG-A = REG-C (caracteres da Constante e da Linha iguais), desvia para a instrução 4270 (E11) para comparar os próximos caracteres.

**** As Instruções 4460 a 4480 retomam a comparação da constante (os primeiros caracteres pesquisados eram iguais, mas não todos).

4460 Restaura em REG-HL, a partir da PILHA, o endereço do BYTE da linha BASIC a partir do qual foi iniciada a comparação com a constante, "Salvo" pela instrução 4260.
4470 Adiciona 1 ao REG-HL, que passa a apontar para o próximo caráter da linha de programa BASIC.
4480 Desvia para a instrução 4250 (E10), re-iniciando a comparação da constante fornecida a partir do próximo caráter da linha BASIC.

**** As Instruções 4490/4500 retornam ao ponto de chamada de VERCHR.

4490 Restaura a PILHA à sua posição original.
4500 RETORNA ao ponto de chamada de VERCHR, com o Indicador CY posicionado em "NC" ou "C" para indicar a existência ou não da constante na linha pesquisada.

20 — ROTINA COPMOV

20.1 - OBJETIVO

O objetivo desta rotina é "Copiar" ou "Mover" uma linha de programa BASIC, ou um conjunto de linhas, de sua posição atual para outra posição neste programa.

Esta nova função é útil, por exemplo, quando queremos inverter a posição de trechos do programa BASIC, ou quando queremos "Duplicar" um trecho de programa, ou ainda quando queremos inserir numa posição determinada uma rotina que já havia sido previamente preparada e que foi "Adicionada" ao final do programa atual pelo comando ".z" do BIT-BASIC.

20.2 - SINTAXE E FUNÇÕES DOS COMANDOS.

20.2.1 - COPIA LINHAS

Sintaxe : .caaa,bbb,ccc
Função : Copia linhas de números aaa até bbb, após a linha de número ccc (DE/ATÉ-Opcional/APÓS).
Exemplos : .c130,150,20 Copia linhas 130 a 150 após linha 20
.c130,20 Copia linha 130 após linha 20

20.2.2 - MOVE LINHAS

Sintaxe : .maaa,bbb,ccc
Função : Move linhas de números aaa até bbb, após a linha de número ccc (DE/ATÉ-Opcional/APÓS).
Exemplos : .m130,20 Move linha 130 para após linha 20
.m130,150,20 Move linhas 130 a 150 após a linha 20

20.3 - CARACTERÍSTICAS

Na CÓPIA de linhas, as Linhas DE/ATÉ permanecem no programa. Na MOVIMENTAÇÃO de linhas, as Linhas DE/ATÉ são eliminadas. Não é necessária a existência no programa das Linha DE/ATÉ, sendo porém obrigatória a presença da Linha-APÓS para que as funções de Cópia ou de Movimentação funcionem com sucesso. No caso de "Movimentação de Linhas", todas as referências a cada uma delas (GOTO/GOSUB) passam a apontar para os seus "Novos Números" correspondentes.

Para ambas as funções, as novas linhas geradas são numeradas de 1 em 1, com numeração iniciando no "Número da Linha-APÓS mais 1", e o processo é automaticamente interrompido quando esta numeração alcança a linha "Seguinte à Linha-APÓS". Por exemplo, para o comando ".c130,150,20" se entre as linhas de números 130 e 150 existem 10 linhas de programa BASIC, e se a linha seguinte à linha de número 20 possui número 25, somente quatro das dez linhas solicitadas serão copiadas (sob os números 21, 22, 23 e 24).

Para "Abrir" um intervalo maior entre as linhas, de forma a poder "Encaixar" a quantidade desejada, pode ser efetuada uma "Falsa Movimentação", renumerando as linhas porém mantendo-as na mesma "Posição Relativa".
No caso do exemplo, poderíamos ter feito ".m25,50,50" com o que as linhas 25 a 50 passariam a possuir numeração a partir de 51, porém permaneceriam no mesmo local. Portanto, após a linha 20 passaríamos a ter um intervalo de "51-20-1=30 números" comportando então 30 "Novas Linhas".

20.4 - ROTINA ASSEMBLER (COPMOV).

**** As instruções 4530 a 4740 obtém os parâmetros DE/ATÉ/APÓS a partir da linha de comando, salvando-os em áreas de trabalho.

```
4530 COPMOV:   LD   (T15),A
4540           CALL PARM
4550           JP   C,RET03
4560           LD   (T10),DE
4570           INC  DE
4580           LD   (T11),DE
4590           CALL RST10
4600           CP   #2C
4610           JP   NZ,RET03
4620           CALL PARM
4630           JP   C,RET03
4640           LD   (T12),DE
4465           CALL RST10
4660           CP   #2C
4670           JR   NZ,H00
4680           PUSH DE
4690           CALL PARM
4700           POP  BC
4710           JP   C,H00
4720           INC  BC
4730           LD   (T11),BC
4740           LD   (T12),DE
```

4530 "Salva" em T15 o "Tipo de Comando" ("c" ou "m").
4540 Obtém Parâmetro Numérico.
4550 Desvia para RET03 se parâmetro incorreto (ERRO).
4560 "Salva" primeiro parâmetro em T10 (Linha-DE).
4570 Adiciona 1 ao REG-DE (Primeiro Parâmetro).
4580 "Salva" REG-DE (Linha-DE + 1) em T11 (Linha-ATÉ +1), para o caso do parâmetro Linha-ATÉ ter sido omitido.
4590 Obtém próximo caráter da linha de comando (em REG-A).
4600 Compara REG-A com "," (#2C).
4610 Desvia para RET03 se o próximo caráter não é "," (ERRO).
4620 Obtém novo parâmetro numérico.
4630 Desvia para RET03 se parâmetro inválido (ERRO).
4640 Salva parâmetro em T12 (Linha-APÓS).
4650 Obtém próximo caráter da linha de comando em (REG-A).
4660 Compara REG-A com "," (#2C).

4670 Desvia para a instrução 4750 (H00) em caso de não existir uma vírgula (#2C) após o segundo parâmetro numérico (assume somente dois parâmetros).
4680 Salva na PILHA o último (Segundo) parâmetro numérico obtido.
4690 Obtém próximo (Terceiro) parâmetro numérico.
4700 Coloca em REG-BC, a partir da PILHA, o segundo parâmetro numérico.
4710 Se o terceiro parâmetro numérico não é válido, desvia para a Instrução 4750 (H00) (assume dois parâmetros).
4720 Soma 1 ao REG-BC.
4730 Coloca REG-BC (Segundo parâmetro +1 = Linha-ATÉ mais 1) em T11 (Linha-ATÉ mais 1).
4740 Coloca REG-DE (Terceiro parâmetro) em T12 (Linha-APÓS).

**** As instruções das linhas 4750 a 4810 "Procuram" a Linha-APÓS no programa BASIC.

```
4750 H00:       LD   (T03),DE
4760            LD   DE,(#F676)
4770 H01:       CALL PXLIN
4780            JR   NC,H02
4790            JP   Z,RET04
4800            JR   H01
4810 H02:       JP   NZ,RET04
```

4750 Coloca REG-DE (número da Linha-APÓS) em T03 (campo utilizado pela rotina PXLIN para comparação).
4760 Carrega em REG-DE o endereço de início do programa (Campo TXTTAB do BASIC).
4770 Obtém "Próxima Linha" do programa BASIC.
4780 Se o "Número da Linha Recuperada" é "Maior ou Igual" a T03, desvia para a Instrução 4810 (H02).
4790 Se condição (C,Z) = "Fim do Programa" (Linha-APÓS não foi encontrada), desvia para a rotina RET04, que efetua retorno normal ao BASIC, sem executar qualquer cópia ou movimentação de linhas.
4800 Desvia para a instrução 4770 (H01) para continuar o processo de busca da Linha-APÓS.
4810 Se a condição (NC,NZ) for alcançada (Linha-APÓS não foi encontrada), desvia para RET04 (idem Instrução 4790).

**** As linhas 4820 a 4910 obtém o "Número da Linha-LIMITE" (linha imediatamente seguinte à Linha-APÓS).

```
4820            CALL PXLIN
4830            JR   NC,H03
4840            LD   BC,#FFF5
4850            JR   H05
4860            DEFS 30
4870 H03:       INC  BC
4880            INC  BC
4890            LD   (H04+2),BC
4900 H04:       LD   BC,(#0000)
4910 H05:       LD   (T13),BC
```

4820 Obtém próxima linha BASIC (linha seguinte à Linha-APÓS).
4830 Se nova linha foi obtida, desvia para a instrução 4870 (H03) (REG-BC contém o Número desta linha).
4840 Se o "Fim do Programa" foi encontrado (a Linha-APÓS era a última linha do programa), assume 65525 (#FFF5) como "Número da Linha-LIMITE" (REG-BC).
4850 Desvia para a instrução 4910 (H05).
4860 BYTES livres para adaptação do BIT-BASIC.
4870 Adiciona 1 ao REG-BC.
4880 Adiciona 1 ao REG-BC, que passa a apontar para o terceiro BYTE da linha (Número da Linha).
4890 Carrega REG-BC no campo de endereço da Instrução seguinte (4900 = H04).
4900 Carrega em REG-BC o "Número da Linha" imediatamente seguinte à Linha-APÓS.
4910 "Salva" REG-BC em T13 (Número da Linha-LIMITE).

**** As linhas 4920 a 4960 preparam as condições para a busca das linhas DE/ATÉ e execução da sua cópia ou movimentação pelas rotinas CML01 e CML02, pertencentes à rotina COPMOV.

```
4920        LD   BC,(T10)
4930        DEC  BC
4940        LD   (T03),BC
4950        XOR  A
4960        LD   (#F414),A
```

4920 Carrega T10 (Linha-DE) em REG-BC.
4930 Subtrai 1 de REG-BC.
4940 Carrega REG-BC em T03 (será utilizado pela rotina PXLIN para comparação).
4950 "Zera" REG-A.
4960 Move #00 para o campo ERRFLAG do BIOS (ERROR FLAG - Indicador de Erro), armazenado no endereço #F414 da RAM. Este campo é utilizado para armazenar o "Número do Erro", em caso de ocorrência de erro, e será utilizado pelo BIT-BASIC para detectar um erro eventual durante os procedimentos de CÓPIA/MOVIMENTACAO de linhas.

**** Na sequência, segue para a rotina CML01 para dar continuidade aos procedimentos da rotina COPMOV.

20.5 - ROTINA ASSEMBLER (CML01)

Esta rotina obtém a "Próxima Linha", entre as Linhas DE/ATÉ, e efetua a Cópia ou Movimento desta linha para o novo local (Número) desejado.

Para que a linha em processo seja "Incluída" no programa BASIC sob outro número, é utilizada uma técnica que deixa para o próprio BASIC as tarefas de "Codificação" e "Inserção" da linha.

Isto é feito para não ser necessário "Tratar" com os complexos procedimentos que executam estas funções.
Por exemplo, a conversão dos comandos e funções BASIC nos TOKENS (Códigos) correspondentes, ou a "Inclusão/Eliminação" de linhas BASIC (com o que as linhas são "realocadas" na memória, pois elas são sempre armazenadas na mesma ordem de sua numeração.)

A técnica utilizada é a de colocar na Tela a linha desejada, já sob o novo número, e acionar a rotina RET01 para "Simular" o acionamento da tecla RETURN sobre esta linha, com o que o BASIC fará a sua captura e tratamento, executando todas as funções "Complexas" anteriormente mencionadas.
Para isto é necessário devolver temporariamente o controle ao BASIC, para cada linha Copiada ou Movimentada, e posteriormente retornar o controle ao BIT-BASIC.
O controle é passado ao BASIC, via rotina RET01, e para que seja devolvido ao BIT-BASIC utilizamos agora o GANCHO instalado no início da ROTINA PRINCIPAL (MAIN-ENTRY) do BASIC (após o tratamento da "Nova Linha" haverá sempre a passagem por este ponto).
Neste GANCHO instalamos um novo desvio para a rotina DESVIO, e "Ligamos" o Indicador T19 para indicar que há um comando de CÓPIA/MOVIMENTAÇÃO sendo processado, com o que a rotina INÍCIO devolverá o controle para a rotina COPMOV (CML02) para a continuação do processo.

As rotinas CML01 e CML02 são repetidas para cada nova linha a ser processada.

```
4980 CML01:     LD   A,#C9
4990            LD   (#FF0C),A
5000            LD   (T19),A
5010            CALL #00B7
5020            JP   C,RET04
```

4980 Coloca #C9 em REG-A.
4990 "Desliga" o GANCHO do início da ROTINA PRINCIPAL do BASIC, instalado a partir do endereço #FF0C, movendo para este endereço o valor #C9, que é uma instrução RET para o Z-80.
Isto é feito para o caso de "Retorno Final" ao BASIC nas próximas Instruções, sendo este GANCHO posteriormente "Religado" se necessário.
5000 "Desliga" o indicador de "Comando de CÓPIA/MOVIMENTAÇÃO em Progresso" (Campo T19).
Idem observações da Instrução 4990.
5010 Efetua chamada à rotina BREAKX do BIOS (verifica CONTROL+STOP).
5020 Caso as teclas CONTROL e STOP estejam sendo simultaneamente pressionadas na passagem do BIT-BASIC pela rotina BREAKX (Indicador CY "Ligado"), o processo de CÓPIA/MOVIMENTAÇÃO é interrompido neste ponto.
Isto é feito para possibilitar ao usuário interromper o processo antes do seu final (via rotina RET04).

**** As Instruções 5030 a 5050 verificam se ocorreu erro no processamento da linha anterior pelo BASIC.

```
5030            LD   A,(#F414)
5040            CP   #00
5050            JP   NZ,RET04
```

5030 Carrega o campo ERRFLAG em REG-A.
5040 Compara ERRFLAG (REG-A) com #00.
5050 Se ERRFLAG diferente de "Zero", o que significa que algum erro ocorreu no processamento da linha BASIC anterior (por exemplo, o BASIC tentou incluir uma nova linha e não havia mais espaço disponível na memória para isto), retorna ao BASIC via rotina RET04 (retorno normal), encerrando a rotina COPMOV.

**** As instruções das linhas 5060 a 5110 "Percorrem" o programa BASIC via rotina PXLIN, até ser encontrada a "Linha-DE menos 1" (é este o número contido no campo T03, utilizado pela rotina PXLIN para comparação, aí colocado pela instrução da linha 4940).
Como será procurada uma linha de número "Maior que Linha-DE menos 1", não é necessário que a Linha-DE esteja presente no programa BASIC para que a função de Cópia ou Movimentação seja executada com sucesso.
A PESQUISA É SEMPRE EFETUADA DESDE O INÍCIO DO PROGRAMA, POIS A CADA INCLUSÃO/EXCLUSÃO DE LINHAS VIA ROTINA COPMOV O PROGRAMA É "REALOCADO" NA MEMÓRIA.

```
5060            LD   DE,(#F676)
5070 H07:       CALL PXLIN
5080            JR   NC,H08
5090            JP   Z,RET04
5100            JR   H07
5110 H08:       JR   Z,H07
```

5060 Carrega em REG-DE o endereço de início do programa BASIC (Campo TXTTAB).
5070 Obtém "Próxima Linha" do programa BASIC.
5080 Se a linha obtida é menor do que a "Linha-DE menos 1", desvia para a Instrução 5110 (H08).
5090 Se "Fim do Programa" (C,Z), ou seja, a "Linha-DE menos 1" não foi encontrada, desvia para a rotina RET04 (retorno normal ao BASIC), interrompendo o processo.
5100 Se as condições anteriores não ocorreram, desvia para a Instrução 5070 (H07) para continuar o processo de busca da Linha-APÓS.
5110 Se Número da linha BASIC igual a T03 (Linha-DE menos 1), desvia para a instrução 5070 (H07) para recuperar a linha seguinte.

**** As Instruções das linhas 5120 a 5150 verificam se a Linha-ATÉ já foi alcançada (encerrando o processo em caso afirmativo).

```
5120            LD   (T01),HL
5130            LD   DE,(T11)
5140            RST  #20
5150            JP   NC,RET04
```

5120 Salva REG-HL no campo T01 (número da linha BASIC a ser Copiada/Movimentada).
5130 Carrega T11 (Número da Linha-ATÉ mais 1) em REG-DE.
5140 Efetua chamada à rotina DCOMPR do BIOS, que compara REG-HL com REG-DE.
5150 Se REG-DE "Não é Maior" que REG-HL, ou seja, se "Linha-ATÉ Mais 1" "Não é Maior" que a "Linha Atual", ou ainda, se a "Linha Atual" é "Maior" que a "Linha-ATÉ", encerra o processo de Cópia ou Movimentação (via rotina RET04).

**** As linhas 5160 a 5210 obtém o número da NOVA LINHA a ser incluída e verificam se este número não ultrapassa a LINHA LIMITE.

```
5160            LD   DE,(T13)
5170            LD   HL,(T12)
5180            INC  HL
5190            LD   (T12),HL
5200            RST  #20
5210            JP   NC,RET04
```

5160 Carrega T13 (Linha-LIMITE) em REG-DE.
5170 Carrega T12 (Linha-APÓS) em REG-HL.
5180 Soma 1 em REG-HL (Linha-APÓS).
Este número será utilizado como NÚMERO DA LINHA NOVA a ser incluída pelo BIT-BASIC.
5190 Retorna REG-HL para T12, que passa a conter o "Número da Linha-APÓS mais i", sendo "i" a quantidade de linhas já processadas.
Com isto passamos a utilizar o campo T12 para armazenar o número da "Linha-NOVA" que está sendo inserida pelo BIT-BASIC.
5200 Compara REG-HL com REG-DE.
5210 Se REG-DE "Não é Maior" que REG-HL, ou seja, se Linha-LIMITE "Não é Maior" que Linha-NOVA, interrompe o processo via rotina RET04.
Com este procedimento, as linhas Copiadas ou Movimentadas (sob nova numeração) utilizam somente o intervalo disponível entre a "Linha-APÓS" e a "Linha Seguinte à Linha-APÓS" ("Linha-LIMITE).

**** As linhas 5220 a 5450 colocam na Tela a linha a ser Copiada/Movimentada, sob o NOVO NÚMERO.
Caso o comando corrente seja de "Movimentacão de Linhas", as instruções 5260 a 5400 fazem com que as outras que apontem para ela (InstruçõesGOTO/GOSUB) passem a apontar para o seu "Novo Número".

```
5220          LD   A,(T15)
5230          LD   (T18),A
5240          CP   #63
5250          JR   Z,H11
5260          PUSH BC
5270          PUSH BC
5280          POP  IX
5290          LD   BC,#54F6
5300          CALL CALL
5310          LD   BC,(T12)
5320          LD   (IX+2),C
5330          LD   (IX+3),B
5340          LD   BC,#54F7
5350          CALL CALL
5360          LD   BC,(T01)
5370          LD   (IX+2),C
5380          LD   (IX+3),B
5390          POP  BC
5400          LD   HL,(T12)
5410 H11:     CALL IMPLIN
5420          LD   HL,(T01)
5430          LD   (T03),HL
5440          XOR  A
5450          LD   (T18),A
```

5220 Carrega o conteúdo do campo T15 ("c" ou "m") em REG-A.
5230 "Liga" o campo T18, que serve para indicar à rotina IMPTELA que "Não é Para Ser Verificada" a condição de "Última Linha Inferior da Tela".
5240 Compara REG-A com "c".
5250 Desvia para a instrução 5410 (H11), caso o comando atual seja para "Cópia de Linhas" (neste caso, instruções GOTO/GOSUB continuam apontando para a linha original).
5260 Coloca REG-BC na PILHA (endereço de início da linha a ser movimentada).
5270 Idem 5260.
5280 Coloca último valor da PILHA em REG-IX (endereço de início da linha a ser movimentada).
5290 Coloca em REG-BC o valor #54F6.
5300 Efetua chamada à rotina do Interpretador BASIC que inicia em #54F6, a qual executa a função de substituir todas as "referências a números de linhas" de um programa BASIC (Instruções GOTO/GOSUB) pelos "endereços destas linhas".
Este procedimento é utilizado pelo Interpretador BASIC para agilizar a execução de um programa (RUN), tornando desnecessária a "procura" da linha para a qual deve ser efetuado o desvio, a cada instrução GOTO ou GOSUB.
Esta técnica é conhecida como "Apontadores Progressivos".

5310 Carrega em REG-BC o campo T12 (novo número da linha a ser movimentada).
5320 Coloca REG-C na terceira posição da linha a ser movimentada.
5330 Coloca REG-B na quarta posição da linha a ser movimentada.
As linhas 5320/5330 colocam o "Número Novo" da linha a ser movimentada no lugar do "Número Antigo" (Número-Atual).
É importante observar que a linha não foi movimentada de sua posição original, ainda que seu número tenha sido alterado (o Interpretador BASIC não faz qualquer verificação a este respeito).
5340 Coloca em REG-BC o valor #54F7.
5350 Efetua chamada à rotina do Interpretador BASIC em #54F7, que "retorna" as referências a "endereços de linhas" para referências a "Números de Linhas".
Como a linha a ser movimentada teve seu "Número-Original" modificado para o "Número-Novo", todas as linhas passarão a apontar para o seu "Número-Novo" após a execução desta rotina.
5360 Carrega em REG-BC o campo T01 (número original da linha a ser movimentada).
5370 Coloca REG-C na terceira posição da linha a ser movimentada.
5380 Coloca REG-B na quarta posição da linha a ser movimentada.
As linhas 5370/5380 retornam o "Número Original" da linha.
NESTE PONTO, A LINHA DO PROGRAMA A SER MOVIMENTADA ESTÁ AINDA EM SUA POSIÇÃO ORIGINAL, COM SEU NÚMERO ORIGINAL, PORÉM AS LINHAS QUE APONTAVAM PARA ESTE NÚMERO JÁ ESTÃO APONTANDO PARA O SEU "NOVO NÚMERO".
Os próximos passos serão incluir esta linha com seu novo número e eliminá-la com seu número antigo.
5390 Restaura em REG-BC o endereço de início da linha a ser movimentada, a partir da PILHA DO SISTEMA.
5400 Restaura em REG-HL o "Novo Número" da linha a ser movimentada (Campo T12).
5410 Efetua chamada à rotina IMPLIN, que fará a colocação na Tela da Linha-NOVA (linha atual do programa BASIC, sob o NOVO NÚMERO contido em REG-HL).
O processo não é interrompido caso a última linha da tela tenha sido atingida, em razão de estar "ligado" o campo T18 (Instrução 5230).
As referências a esta linha (GOTO/GOSUB) já estão atualizadas para este "Novo Número", conforme descrito anteriormente.
5420 Restaura em REG-HL o "Número Antigo" da Linha-Atual (T01), "Salvo" pela Instrução 5120.
5430 Carrega REG-HL (T01) no campo T03.
As Instruções 5420/5430 "Recolocam" o "Número Antigo" da Linha-Atual no campo T03, para obter a "Próxima Linha" quando da passagem seguinte pela rotina PXLIN.
(Este procedimento é necessário porque a rotina IMPLIN altera o valor de T03.)
5440 "Zera" REG-A.
5450 Desliga "Contorno" da verificação de "Última Linha" da Tela (Campo T18).

**** As linhas 5460 a 5510 posicionam o CURSOR sobre a LINHA-NOVA, e passam o controle ao BASIC, "Simulando" o pressionamento da tecla RETURN sobre esta linha.

```
5460            LD    HL,#F3DC
5470            DEC   (HL)
5480            LD    A,#C3
5490            LD    (#FF0C),A
5500            LD    (T19),A
5510            JP    RET01
```

5460 Carrega em REG-HL o endereço da "Posição Vertical Atual" do CURSOR (CSRY - #F3DC), que está posicionado na linha da Tela imediatamente seguinte à Linha-NOVA recém apresentada (a rotina IMPL deixa o CURSOR nesta posição).

5470 Subtrai 1 do endereço apontado por REG-HL (CSRY - #F3DC), retrocedendo o CURSOR de uma linha, o qual fica posicionado sobre a Linha-NOVA, recém colocada na Tela.

5480 Coloca o valor #C3 em REG-A.

5490 "Liga" o GANCHO instalado no endereço #FF0C, acionado no início da ROTINA-PRINCIPAL do BASIC, colocando o caráter #C3 em sua primeira posição (código da instrução JUMP = DESVIO do Z-80).
O endereço para o qual será efetuado o NOVO DESVIO, após a conclusão do processamento da Linha-NOVA pelo BASIC, já foi previamente armazenado nos dois BYTES seguintes (#FF0D/FF0E) pela rotina que efetua o carregamento do programa BIT-BASIC na memória.
Este endereço é #FFEB e correspondente ao Ponto de Entrada DV01 da rotina DESVIO, que fará novamente a interceptação do BASIC, com retorno do controle ao BIT-BASIC.

5500 "Liga" o campo T19, que indicará à rotina INICIO que está sendo processado um comando de CÓPIA/MOVIMENTAÇÃO.

5510 Devolve o controle ao BASIC via rotina RET01, que fará a "Simulação" do pressionamento da tecla RETURN sobre a Linha-ATUAL, recém colocada na Tela, e posteriormente retornará o controle ao BIT-BASIC.
O retorno à rotina COPMOV (CML02) é efetuado via rotina INICIO, que identifica por intermédio do campo T19 que há um comando de Cópia ou Movimentação "Sendo Executado".
A rotina INICIO desvia então para a rotina CML02, a seguir descrita, para continuação do processo.

20.6 - ROTINA ASSEMBLER (CML02)

Esta rotina obtém o controle a partir do GANCHO instalado no início da ROTINA-PRINCIPAL (MAIN-ENTRY) do BASIC, quando o desvio ali instalado estiver "Ligado" pelos procedimentos já descritos para a rotina CML01 (Instruções 5480/5510).
Uma de suas funções é "Efetuar a eliminação da linha BASIC original", no caso de comando de MOVIMENTAÇÃO de linhas.
A outra função é "Retornar à rotina CML01 para o processamento da linha BASIC seguinte".

**** As instruções 5540 a 5560 devolvem o controle à rotina CML01, caso o comando em processo seja CÓPIA de linhas.

```
5540 CML02:    LD   A,(T15)
5550           CP   #63
5560           JR   Z,CML01
```

5540 Coloca o campo T15 (tipo de Comando, "c" ou "m").
5550 Compara REG-A com "c" (#63).
5560 Caso o campo T15 contenha "c", desvia para a rotina CML01 para continuação do processo de CÓPIA de linhas.

**** Como no caso de "Movimentação de Linhas" a rotina CML02 recebe o controle duas vezes a cada linha BASIC processada (a primeira após a inserção da Linha-NOVA, e a segunda após a "Eliminação" da Linha-ORIGINAL), as Instruções 5570 a 5600 verificam se a passagem pela rotina CML02 está sendo feita pela primeira ou segunda vez.
A técnica empregada utiliza um "FLIP-FLOP" ("LIGA-DESLIGA") sobre o campo T14 (originalmente "Desligado" - #00).

```
5570           LD   A,(T14)
5580           XOR  #FF
5590           LD   (T14),A
5600           JP   Z,CML01
```

5570 Carrega T14 em REG-A.
5580 Efetua a comparação de REG-A com o valor #FF, BIT a BIT, na modalidade "XOR", colocando o resultado em REG-A.
Esta instrução "Inverte" os valores "0" ou "1" dos BITS de REG-A, que conterá alternadamente os valores #00 e #FF a cada nova passagem por esta instrução.
5590 Retorna REG-A "Invertido" para T14.
5600 Caso o resultado da operação XOR sobre REG-A resulte em #00 (aqui o Indicador Z é "Ligado"), desvia para a rotina CML01, pois isto indica que esta é a "Segunda" passagem para a linha BASIC atual (a "Eliminação" da Linha-ORIGINAL já foi efetuada).
Caso o resultado da operação XOR seja #FF, segue para a Instrução seguinte, para "Eliminar" a Linha-ORIGINAL.

**** As instruções 5610 a 5660 fazem com que o BASIC efetue a "Eliminação" da Linha-ANTERIOR, colocando na Tela somente o número desta linha (seguido de "Brancos"); e simulando o acionamento da tecla RETURN sobre este número.

```
5610          LD   A,#20
5620          RST  #18
5630          LD   HL,(T01)
5640          LD   BC,#3412
5650          CALL CALL
5660          JP   RET01
```

5610 Coloca #20 ("Branco") em REG-A.
5620 Apresenta o caráter "Branco" na tela.
Neste momento o CURSOR está posicionado na primeira posição da linha seguinte à Linha-NOVA recém Inserida (a simulação da tecla RETURN sobre esta linha deixa o CURSOR nesta posição).
Este caráter "Branco" é apresentado antes do número da linha a ser "Eliminada" para proporcionar um efeito visual de "Deslocamento".
5630 Carrega T01 em REG-HL (número da linha a ser "Eliminada", "Salvo" pela instrução 5120).
5640 Coloca o endereço #3412 em REG-BC.
5650 Coloca na Tela o número da Linha-ORIGINAL (veja item II-13.4, instrução 2320).
5660 Devolve o controle ao BASIC, via rotina RET01, que fará a "Simulação" do pressionamento da tecla RETURN sobre o número de linha recém colocado na Tela, com o que esta linha será eliminada do programa pelo BASIC.
Após a eliminação, o BASIC irá passar pela ROTINA-PRINCIPAL, que efetuará chamada ao GANCHO em #FF0C, onde já se encontra "Ligado" o desvio para a rotina DESVIO do BIT-BASIC, a qual devolverá o controle à Instrução 5540 desta rotina (CML02), que por sua vez dará continuidade ao processo conforme já descrito.

21 - ROTINA SINTAXE

21.1 - OBJETIVO

A rotina SINTAXE atende a dois objetivos:

1) "Simplificar a Sintaxe" para vários comandos BASIC, facilitando ainda mais a sua utilização pelos usuários MSX.
2) Acionar programas BASIC "de uma linha", previamente preparados e armazenados em uma TABELA INTERNA do BIT-BASIC, por intermédio de "Sintaxe Simplificada".
   Podemos ter, portanto, vários programas BASIC "Utilitários" já armazenados na memória RAM, prontos para execução sem que seja necessário carregá-los a partir de "Disco ou Fita", e sem destruir o programa BASIC com o qual você está trabalhando.

Assim, por exemplo, para executar o comando FILES basta digitar ".f" e acionar a tecla RETURN, e para mostrar o conteúdo da memória em hexadecimal, entre os endereços #B000 e #B010, basta digitar ".hB000,B010".

O USUÁRIO PODE MODIFICAR, A SEU CRITÉRIO, AS REGRAS DE SIMPLIFICAÇÃO DE SINTAXE PARA COMANDOS BASIC, E PODE MODIFICAR OU INCLUIR PROGRAMAS BASIC "DE UMA LINHA" NA TABELA BIT-BASIC.

21.2 - SINTAXE E FUNÇÕES DOS COMANDOS.

Veja ÍTEM II-23, "TABELA DE SINTAXE".

21.3 - CARACTERÍSTICAS

Esta rotina é acionada quando a primeira posição do comando BIT-BASIC contém o caráter ".", seguido de um ou dois caracteres que identificam o comando ou programa BASIC a ser executado.
O comando BIT-BASIC é "Transformado" em comando BASIC correspondente, de acordo com a sintaxe por ele requerida, ou em um "Programa BASIC" previamente armazenado, o qual é então passado ao Interpretador BASIC para execução.
Para efetuar a conversão do comando BIT-BASIC para a sintaxe do BASIC é utilizada uma TABELA que "Dirige" a rotina SINTAXE neste procedimento.
Esta tabela contém "Caracteres de Controle" (intermeados a valores constantes) que indicam em cada ponto se os "Próximos Caracteres" da linha BASIC devem ser obtidos a partir da TABELA ou da "Linha de comando BIT-BASIC".
A passagem ao Interpretador BASIC pode ser efetuada a partir da TELA ("Simulando" o pressionamento da tecla RETURN sobre a linha via rotina RET01) ou a partir do BUFFER INTERNO, fazendo com que o comando/programa gerado NÃO APAREÇA NA TELA.
A opção de APARECER/NÃO-APARECER na tela pode ser selecionada pelo usuário.

Veja o ítem II-23 para maiores detalhes.

21.4 - ROTINA ASSEMBLER (SINTAXE)

**** As Instruções 5690 a 5740 procuram na TABELA o caráter da linha de comando BIT-BASIC seguinte ao "."

```
5690 SINTAXE   CALL ENDE
5700 I01:      LD   BC,T21
5710           CP   #FA
5720           JR   NZ,I02
5730           DEC  HL
5740           LD   DE,ERR
```

5690 Efetua chamada à rotina ENDE, que pesquisará a TABELA procurando o primeiro ou os dois primeiros caracteres da linha de comando BIT-BASIC após o ".".
5700 Carrega em REG-BC o endereço do campo T21 (Área de Trabalho do BIT-BASIC onde será montada a linha a ser passada ao BASIC para execução, via TELA ou via BUFFER DO BASIC).
5710 Compara REG-A com #FA (Fim da TABELA encontrado).
5720 Se a rotina ENDE não colocou #FA em REG-A, o que indica que o comando solicitado foi encontrado na TABELA, desvia para a Instrução 5750 (I02), para continuação do processo.
5730 Subtrai 1 de REG-HL ("Retorna" para caráter anterior do comando).
5740 Carrega em REG-DE o endereço da linha da TABELA encarregada de estabelecer a condição de ERRO DE SINTAXE. (Repete na Tela a linha originalmente digitada, substituindo o primeiro caráter "." por ":")

**** As instruções 5750 a 6010 analisam cada caráter da linha da TABELA, um a um, desviando para a Instrução encarregada de executar a função correspondente.
REG-DE contém o endereço da "Entrada" na TABELA.
REG-BC contém o endereço da "Área de Trabalho" BIT-BASIC.
REG-HL contém o endereço do "Comando BIT-BASIC" fornecido pelo usuário.

```
5750 I02:      LD   A,(DE)
5760           INC  DE
5770           CP   #00
5780           JP   Z,I09
5790           CP   #FF
5800           JR   Z,I05
5810           CP   #FE
5820           JR   Z,I07
5830           CP   #FC
5840           JR   Z,I08
5850           CP   #F8
5860           JR   NZ,I03
5870           LD   A,(HL)
5880           LD   (T17),A
5890           INC  HL
5900           JR   I05
```

```
5920 I03:      CP   #F9
5930           JR   NZ,I04
5940           LD   A,(T17)
5950 I04:      CP   #FB
5960           JR   Z,I4A
5970           LD   (BC),A
5980           INC  BC
5990           JR   I02
6000 I4A:      LD   (T20),A
6010           JR   I02
```

5750 Coloca em REG-A o próximo caráter da entrada na TABELA.
5760 Adiciona 1 em REG-DE (passa a apontar para o próximo caráter na TABELA).
5770 Compara REG-A com #00.
5780 Se REG-A = #00 (fim da entrada na TABELA), desvia para a instrução 6220 (I09), que fará o retorno ao BASIC.
5790 Compara REG-A com #FF.
5800 Se REG-A = #FF, desvia para 6020 (I05).
5810 Compara REG-A com #FE.
5820 Se REG-A = #FE, desvia para 6120 (I07).
5830 Compara REG-A com #FC.
5840 Se REG-A = #FC, desvia para 6190 (I08).
5850 Compara REG-A com #F8.
5860 Se REG-A diferente de #F8, desvia para 5920 (I03).
5870 Se REG-A = #F8, carrega o BYTE apontado por REG-HL (caráter atual da linha de comando BIT-BASIC) em REG-A.
5880 "Salva" REG-A no campo de trabalho T17.
5890 Soma 1 em REG-HL, passando a apontar para o próximo caráter de linha de comando.
5900 Desvia para 6020 (I05), para processar o próximo caráter.
5920 Compara REG-A com #F9.
5930 Se REG-A diferente de #F9, desvia para 5950 (I04).
5940 Se REG-A = #F9, recupera em REG-A o caráter salvo em T17 quando da execução do último caráter #F8.
5950 Compara REG-A com #FB.
5960 Se REG-A igual a #FB, desvia para 6000 (I4A).
5970 Se REG-A diferente de #F9, o que indica que não é nenhum dos caracteres de controle definidos, o caráter obtido a partir da linha de comando é colocado na ÁREA DE TRABALHO, na posição apontada por REG-BC.
5980 Soma 1 em REG-BC, apontando para o próximo caráter da Área de Trabalho.
5990 Desvia para a instrução 5750 (I02), para obtenção e processamento do próximo caráter da entrada na TABELA.
6000 Se REG-A contém #FB, este valor é colocado no Campo de Trabalho T20 do BIT-BASIC.
A presença deste caráter na TABELA indica que a linha de comando a ser processada, no seu "Formato Final", deve ser apresentada na TELA. Se #FB não está presente na TABELA, a linha não será apresentada.

**** As linhas 6020 a 6100 colocam na "Área de Trabalho" os próximos caracteres da linha de comando BIT-BASIC (apontada por REG-HL), até que seja encontrado um caráter #00 (Fim da Linha) ou um caráter #2C ("," = Fim de Parâmetro).

```
6020 I05:       LD   A,(HL)
6030 I06:       CP   #00
6040            JR   Z,I02
6050            INC  HL
6060            CP   #2C
6070            JR   Z,I02
6080            LD   (BC),A
6080            INC  BC
6100            JR   I05
```

6020 Coloca em REG-A o próximo caráter da linha de comando BIT-BASIC, apontado por REG-HL.
6030 Compara REG-A com #00.
6040 Se REG-A = #00 (Fim de Comando), desvia para a Instrução 5750 (I02), para verificar qual o próximo procedimento comandado pela entrada na TABELA.
6050 Soma 1 em REG-HL (aponta próximo caráter Linha de Comando).
6060 Compara REG-A com #2C.
6070 Se REG-A = #2C ("," = Fim de Parâmetro), desvia para a Instrução 5750 (I02).
6080 Coloca caráter da linha de comando na Área de Trabalho.
6090 Aponta para próximo caráter da Área de Trabalho.
6100 Desvia para a instrução 6020 (I05), para obter e processar o próximo caráter da linha de comando, até ser encontrado #00 ou #2C.

**** As instruções das linhas 6120 a 6170 providenciam a pesquisa do "Número da Primeira/Última Linha do Programa BASIC".

```
6120 I07:       LD   A,(HL)
6130            CP   #63
6140            CALL Z,PRIMLIN
6150            CP   #66
6160            CALL Z,ULTLIN
6170            JR   I05
```

6120 Carrega em REG-A o caráter atual da linha de comando BIT-BASIC.
6130 Compara REG-A com #63 ("c").
6140 Se REG-A = "c", procura a primeira linha do programa BASIC e coloca seu número na Área de Trabalho.
6150 Compara REG-A com #66 ("f").
6160 Se REG-A = "f", procura a última linha do programa BASIC e coloca seu número na Área de Trabalho.
6170 Desvia para a Instrução 6020 (I05).

**** As instruções 6190 a 6210 retornam o controle ao BASIC, caso o fim da linha decomando tenha sido alcançado durante o processamento de um caráter de controle #FC da TABELA.

```
6190 I08:      LD   A,(HL)
6200           CP   #00
6210           JP   NZ,I02
```

6190 Coloca em REG-A o caráter atual da linha de comando.
6200 Compara REG-A com #00.
6210 Se o "Fim de comando" não foi alcançado, desvia para a instrução 5750 (I02).

**** As Instruções 6220 a 6380 fazem com que o BASIC execute a linha de comando gerada na Área de Trabalho, verificando se ela deve ou não ser apresentada na Tela.
Em caso afirmativo, a linha é colocada na Tela e é entregue ao Interpretador BASIC para processamento com a simulação do acionamento da tecla RETURN (Rotina RET01).
Caso a linha não deva ser apresentada na Tela, ela é colocada no BUFFER-BASIC (#F55E) e o controle é retornado ao Interpretador BASIC no ponto " Após a rotina PINLIN" (via rotina RET03).

```
6220 I09:      LD   (BC),A
6230           LD   BC,(T20)
6240           LD   (T20),A
6250           CP   C
6260           LD   HL,T21
6270           LD   DE,#F55E
6280           LD   BC,#00FA
6290           LDIR
6300           LD   (DE),A
6310           JP   Z,RET03
6320           LD   HL,#F55E
6330 I10:      LD   A,(HL)
6340           INC  HL
6350           CP   #00
6360           JP   Z,RET01
6370           RST  #18
6380           JR   I10
```

6220 Coloca #00 (REG-A) na Área de Trabalho, após a última posição da linha de comando gerada (para indicar o "Fim" desta linha).
6230 Carrega o Campo T20 em REG-C (utilizado REG-BC porque não há Instrução Z-80 para colocar uma só posição de memória em REG-C).
6240 Coloca #00 em T20 (REG-A), para "Desligar" este indicador de presença do caráter #FB na Tabela.

6250 Compara o conteúdo de T20 (REG-C) com #00 (REG-A).
6260 Carrega o endereço de início da Área de Trabalho (T21) em REG-HL.
6270 Carrega o endereço de início do BUFFER BASIC (#F55E) em REG-DE.
6280 Carrega #FA (250) em REG-BC.
6290 Copia 250 BYTES da ÁREA DE TRABALHO para o BUFFER BASIC.
6300 Coloca #00 (REG-A) no BUFFER BASIC, após a última posição copiada, para indicar "Fim do Comando" em caso de "Linha muito Grande".
6310 Caso o Campo T20 contenha #00 (resultado do teste efetuado na Instrução 6250), o que indica que a linha a ser processada pelo Interpretador BASIC "NÃO DEVE" ser apresentada na Tela, desvia para a Rotina RET03 (após a chamada à rotina PINLIN do BIOS).
6320 Carrega o valor #F55E em REG-HL (endereço de início do BUFFER-BASIC).
6330 Coloca em REG-A um caráter da linha de comando a ser processada, a partir do BUFFER BASIC.
6340 Aponta para próxima posição do BUFFER BASIC.
6350 Compara REG-A com #00.
6360 Se REG-A igual a #00 (fim da linha de comando alcançado), desvia para a rotina RET01, que fará a "Simulação" do pressionamento da tecla RETURN sobre esta linha colocada na Tela.
6370 Coloca caráter contido em REG-A na Tela.
6380 Desvia para a Instrução 6330 (I10) para processar o próximo caráter da linha contida no BUFFER BASIC.

21.5 ROTINA ASSEMBLER (LISTLIN)

**** As instruções das linhas 6410 a 6430 efetuam o encaminhamento de comandos para "Listar Linhas", utilizando a rotina LISTLIN já descrita, porém sem efetuar a pesquisa na TABELA.
LISTLIN é acionada pela rotina INICIO, quando é detectado um caráter "Numérico" após o Primeiro caráter (".") na linha de comando BIT-BASIC.

```
6410 LISTLIN:  LD   DE,TN
6420           LD   A,#00
6430           JP   I01
```

6410 Carrega em REG-DE o endereço de início da entrada na TABELA, que contém as constantes e códigos de controle necessários para a colocação na Tela de um comando "LIST" do BASIC (utilizando a própria rotina SINTAXE já descrita).
6420 Carrega #00 em REG-A (para que a comparação com #FA da Instrução 5710 resulte negativa).
6430 Desvia para a Instrução 5700 (I01), pertencente à rotina SINTAXE.

21.6 - ROTINA ASSEMBLER (ENDE)

Esta rotina pesquisa a TABELA armazenada a partir do endereço #600E, buscando encontrar uma "Entrada" nesta tabela que corresponda aos caracteres existentes após o "." na linha de comando BIT-BASIC.
Esta Entrada conterá as constantes e regras para a conversão de sintaxe do comando BIT-BASIC para o comando BASIC equivalente.

CONDIÇÕES DE ENTRADA :

REG-HL - Endereço do caráter seguinte ao "." na linha de comando.

CONDIÇÕES DE SAÍDA :

REG-DE - Endereço da entrada na TABELA correspondente aos dois caracteres após o "." no comando BIT-BASIC.
REG-A - Igual a #FA indica "Comando não Encontrado".
Diferente de #FA indica "Comando Encontrado".

**** As Instruções 6450 a 6510 preparam o início da pesquisa.

```
6450 ENDE:      LD   B,A
6460            INC HL
6470            LD   A,(HL)
6480            LD   C,A
6490            LD   DE,TABELA
6500            LD   A,(DE)
6510            INC  DE
```

6450 Coloca primeiro caráter após o "." em REG-B.
6460 Aponta para próxima posição linha de comando.
6470 Coloca segundo caráter após o "." em REG-A.
6480 Coloca REG-A em REG-C.
6490 Coloca em REG-DE o endereço da primeira posição da TABELA.
6500 Coloca em REG-A o primeiro caráter da primeira posição da TABELA.
6510 Aponta para a segunda posição da TABELA.

**** As instruções 6520 a 6600 procuram na tabela o primeiro caráter desejado (contido em REG-B).

```
6520 I20:       CP   B
6530            JR   Z,I22
6540 I21:       CP   #FA
6550            RET  Z
6560            CP   #00
6570            LD   A,(DE)
6580            INC  DE
6590            JR   Z,I20
6600            JR   I21
```

6520 Compara REG-A com REG-B (caráter da Tabela com caráter da Linha de Comando).
6530 Se REG-A = REG-B, desvia para a Instrução 6610 (I22) (a entrada atual na Tabela contém o caráter pesquisado).

**** As instruções das linhas 6540 a 6600 "Pulam" a entrada atual da Tabela, até o início da próxima entrada (já que o caráter procurado não foi encontrado na entrada atual).

6540 Compara REG-A com #FA.
6550 Se REG-A = #FA (Fim da Tabela foi atingido), Retorna ao ponto de chamada de ENDE.
(O valor #FA em REG-A indica que nenhuma Entrada da TABELA possui os dois primeiros caracteres iguais aos dois caracteres após o "." no comando BIT-BASIC.)
6560 Compara REG-A com #00.
6570 Carrega em REG-A o próximo caráter da Tabela.
6580 Soma 1 ao REG-DE (próximo caráter da Tabela).
6590 Se REG-A = #00 (teste efetuado na linha 6560), o que indica que o "Fim de Uma Entrada na Tabela" foi atingido, desvia para 6520 (I20) para comparar próxima entrada.
6600 Desvia para 6540 (I21) para pular os próximos caracteres da Entrada atual, até encontrar #00 ou #FA.

**** As Instruções 6610 a 6680 verificam se a "Identificação" do Comando BIT-BASIC após o "." deve conter um ou dois caracteres.
Caso deva conter apenas um, retorna ao ponto de chamada de ENDE, pois a primeira posição após o "." coincide com a primeira posição da Entrada Atual da TABELA.
Caso deva conter dois caracteres, é verificada a coincidência ou não do segundo caráter após o "." com o segundo caráter da Entrada na TABELA.

```
6610 I22:      LD   A,(DE)
6620           CP   #20
6630           INC  DE
6640           RET  Z
6650           CP   C
6660           JR   NZ,I21
6670           INC  HL
6680           RET
```

6610 Coloca em REG-A a segunda posição da Entrada Atual da TABELA (apontada por REG-DE).
6620 Compara REG-DE com #20
6630 Aponta para próxima posição da TABELA.
6640 Caso a segunda posição da Entrada Atual da Tabela seja "Branco" (#20), retorna ao ponto de chamada de ENDE, pois o primeiro caráter é já coincidente e é necessário apenas um caráter para identificar o comando (#20 indica isto).
6650 Compara a segunda posição após o "." com a segunda posição da Entrada na TABELA.

6660 Caso as segundas posições não coincidam, desvia para a Instrução 6540 (I21) para continuação da pesquisa nas próximas Entradas da TABELA.
6670 Aponta para o próximo caráter da linha de comando.
6680 Retorna ao ponto de chamada de ENDE com os dois caracteres coincidentes.

21.7 - ROTINA ASSEMBLER (PRIMLIN/ULTLIN)

Estas rotinas procuram a primeira/última linhas do programa BASIC corrente e colocam na Área de Trabalho os números correspondentes.
Elas são utilizadas na substituição das constantes "c" e "f" nos comandos BIT-BASIC pelos números da primeira/última linhas.

**** As instruções 6710 a 6750 "Salvam" os Registradores e obtém o número da primeira linha do programa BASIC.

```
6710 PRIMLIN:  PUSH HL
6720           PUSH DE
6730           PUSH BC
6740           LD   DE,(#F676)
6750           CALL PXLIN
```

6710 "Salva" REG-HL na PILHA.
6720 "Salva" REG-DE na PILHA.
6730 "Salva" REG-BC na PILHA.
6740 Carrega em REG-DE o camo TXTTAB (Endereço de Inicio do programa BASIC).
6750 Obtém "Próxima Linha" BASIC (Primeira).

**** As instruções das linhas 6770 a 6860 colocam o número da linha na Área de Trabalho, e restauram os Registradores. Para isto, é utilizada a rotina do BIOS a partir do endereço #36DB, que faz a conversão do número armazenado em dois BYTES (Formato Binário) para o formato ASCII (um BYTE para cada dígito).

```
6770 I30:      LD   (#F7F8),HL
6780           POP  HL
6790           LD   BC,#0000
6800           CALL #36DB
6810           LD   B,H
6820           LD   C,L
6830           POP  DE
6840           POP  HL
6850           INC  HL
6860           RET
```

6770 Coloca o Número da Linha (armazenado em REG-HL pela rotina PXLIN) nos endereços #F7F8/#F7F9.
6780 Coloca em REG-HL o endereço da Área de Trabalho do BIT-BASIC (armazenado na PILHA pela Instrução 6730).
6790 Coloca o valor #0000 em REG-BC.

6800 Efetua chamada à rotina do BIOS no endereço #36DB, cuja função é "Converter para ASCII o valor Binário armazenado a partir do endereço #F7F8 (que contém o Número da Linha), colocando cada dígito convertido no endereço apontado por REG-HL (que aponta para a Área de Trabalho)".
6810 Coloca REG-H em REG-B.
6820 Coloca REG-L em REG-C.
As Instruções 6810/6820 colocam REG-HL em REG-BC, que volta a conter o Endereço da Área de Trabalho, porém apontando para a posição imediatamente seguinte ao Número da Linha no formato ASCII.
6830 Recupera REG-DE da PILHA.
6840 Recupara REG-HL da PILHA.
6850 Soma 1 em REG-HL (aponta para caráter seguinte da Linha de Comando BIT-BASIC).
6860 RETORNA ao ponto de chamada de PRIMLIN/ULTLIN.

**** As instruções 6890 a 6980 "Salvam" os Registradores e obtém o número da última linha do programa BASIC.

```
6890 ULTLIN:    PUSH HL
6900            PUSH DE
6910            PUSH BC
6920            LD   DE,(#F676)
6930 I40:       LD   (T16),HL
6940            CALL PXLIN
6950            JR   NC,I40
6960            JR   NZ,I40
6970            LD   HL,(T16)
6980            JR   I30
```

6890 "Salva" REG-HL na PILHA.
6900 "Salva" REG-DE na PILHA.
6910 "Salva" REG-BC na PILHA.
6920 Carrega em REG-DE o campo TXTTAB (Endereço de Inicio do programa BASIC).
6930 "Salva" REG-HL ("Endereço da Linha BASIC Atual") em T16.
6940 Obtém "Próxima Linha" do programa BASIC.
6950 Se "Fim do Programa" não foi alcançado, desvia para 6930 (I40) para obter a próxima linha (esta condição é representada por C,Z).
6960 Se "Fim do Programa" não foi alcançado, desvia para 6930 (I40) para obter a próxima linha (Idem).
6970 Carrega em REG-HL o campo T16 ("Número da Última Linha BASIC"), aí colocado pela instrução 6930.
6980 Desvia para a instrução 6770 (I30) para colocação do número da linha na Área de Trabalho e retorno ao ponto de chamada de ULTLIN.

22 - ÁREAS DE TRABALHO

O BIT-BASIC coloca os Campos de Trabalho T01 a T21 a partir do endereço #7B00.
Estes campos são utilizados em diversos pontos do programa e os seus conteúdos são os seguintes :

#7B00 T01 (02) - Número da primeira linha do programa BASIC listada na Tela.
#7B02 T02 (20) - Tabela com 20 BYTES, onde são guardados os números das linhas correspondentes aos "Índices" de "0" a "9" (dois BYTES para cada Índice).
#7B16 T03 (02) - Número da linha utilizado para comparação pela rotina PXLIN.
O BIT-BASIC mantém neste campo a "Linha Seguinte à Última Linha Apresentada na Tela".
#7B18 T04 (02) - Quantidade de linhas a retroceder.
#7B1A T05 (06) - Constante "CLEAR", utilizada na rotina NOVOBAS.
#7B20 T06 (02) - "Salva" o campo TXTTAB (endereço de inicio do programa BASIC na memória).
#7B22 T07 (01) - Indica à rotina que "Lista Página" se está sendo executado um comando de "Pesquisa de Constante".
#7B23 T08 (01) - Caráter "=", utilizado pela rotina de "Pesquisa de Constante".
#7B38 T09 (21) - Constante a ser pesquisada na rotina VERCHR.
#7B39 T10 (02) - Número da Linha-DE (Cópia/Movimentação Linhas).
#7B3B T11 (02) - Número da Linha-ATÉ + 1 (Cópia/Movimentação).
#7B3D T12 (02) - Número da Linha-APÓS (Cópia/Movimentação).
#7B3F T13 (02) - Número da Linha-LIMITE (Cópia/Movimentação).
#7B41 T14 (01) - Indica "Primeira ou Segunda Passagem" pela rotina CML02 (Liga/Desliga), em comandos ".m"
#7B42 T15 (01) - "Tipo de comando" ("c" ou "m")
#7B43 T16 (02) - Endereço de início da "Linha BASIC Atual".
#7B45 T17 (01) - Salva caráter para utilização posterior.
#7B46 T18 (01) - Indica à rotina IMPTELA se é para ser verificada a condição de "Última Linha Inferior da Tela".
#7B47 T19 (06) - Indica à rotina INICIO se está sendo executado comando de "Cópia ou Movimentação de Linhas".
#7B4D T20 (1) - Indica se Linha de comando deve ou não ser apresentada na Tela.
#7F00 T21 (255) - Área de Trabalho do BIT-BASIC para conversão de Sintaxe dos comandos ".".

23 - TABELA DE SINTAXE

23.1 - OBJETIVO.

O objetivo desta Tabela é dirigir a rotina SINTAXE na tarefe de "Converter o Comando BIT-BASIC Digitado pelo Usuário" para a "Sintaxe Requerida pelo BASIC".
Por exemplo, se o usuário digitar o comando BIT-BASIC ".f" e teclar RETURN, este será convertido no comando "files" antes de ser entregue ao BASIC para execução.
A Linha de Comando a ser executada pelo BASIC pode ou não ser apresentada na Tela, conforme opção do usuário.

Cada "Entrada" (Linha) da Tabela corresponde a um "Comando" diferente, identificado pelas duas primeiras posições.
Cada Entrada da Tabela é encerrada pelo caráter #00.
O "Fim da Tabela" é identificado pelo caráter #FA.

Uma Entrada pode efetuar a Conversão de Sintaxe para um "Comando BASIC" determinado ou para um "Programa BASIC" previamente armazenado.
Podemos ter, portanto, vários programas BASIC "Utilitários" prontos para execução, sem necessidade de carregá-los de disco ou fita.

O BIT-BASIC JÁ POSSUI UMA TABELA PRÓPRIA, PORÉM A TABELA DE SINTAXE É MODIFICÁVEL PELO USUÁRIO, QUE PODE ALTERAR OU INCLUIR NOVAS ENTRADAS, A SEU CRITÉRIO.
ACOMPANHA O BIT-BASIC UM PROGRAMA BASIC (BITBAS.TAB) QUE SERVE PARA MODIFICAR A TABELA DE SINTAXE.

23.2 - SINTAXE DOS COMANDOS BIT-BASIC.

O comando BIT-BASIC é identificado por um caráter "." na primeira posição da linha digitada pelo usuário, seguida por um ou dois caracteres que identificam o tipo do comando (conforme determinado pela TABELA), seguidos por sua vez de um ou mais parâmetros separados por vírgula.

```
.        = Comando Simplificação de Sintaxe.
x ou xx = Código do Comando BIT.BASIC.
...,... = Um ou mais parâmetros, separados por ","
```

A conversão de sintaxe para "Comandos BASIC" será mostrada por intermédio de exemplos, apresentando o "Comando BIT-BASIC" e a "Linha BASIC" gerada correspondente.
Para os "Programas Simplificados" será mostrado o "Comando BIT-BASIC" e a "Função" correspondente. A listagem dos programas gerados pode ser vista no ítem 23.5 (Listagem do Programa BITBAS.TAB), ou executando os comandos de forma "Explícita" (atualizando a Tabela com [$] nas linhas do BITBAS.TAB).

23.3 - COMANDOS BIT-BASIC X COMANDOS BASIC.

```
.f          =  files
.fa         =  files"a:"
.fb,pgm1    =  files"b:pgm1"
.fb,pg*.* =  files"b:pg*.*"

.la,pgm1    =  load"a:pgm1"
.lb,pgm2    =  load"b:pgm2"
.lc,pgm3    =  load"cas:pgm3"

.sa,pgm1     =  save"a:pgm1"
.sb,pgm1,a =  save"a:pgm1",a
.sc,pgm3     =  save"cas:pgm3"

.kb,pgm1     =  kill"b:pgm1"

.na,pgm1,pgn = name"a:pgm1"as"a:pgn"

.ga,pgm2     =  merge"a:pgm2"

.bla,pgm3             = bload"a:pgm3"
.blb,pgm3,r           = bload"b:pgm3",r
.bla,pgm4,r,C000      = bload"a:pgm4",r,&hC000
.blc,pgm5             = bload"cas:pgm5"

.bsa,pgm5,c000,c300
                      = bsave"a:pgm5",&hc000,&hc300
.bsb,pgm6,aab3,b5df,b400
                      = bsave"a:pgm6",&haab3,&hb5df,&hb400
.bsc,pgm7,aaaa,bbbb,cccc
                      = bsave"cas:pgm7",&haaaa,&hbbbb,&hcccc

.v           =  renum
.v50         =  renum50
.v50,30      =  renum50,30
.v50,30,5    =  renum50,30,5

.ucd00        = defusr=&hcd00:a=usr(0)

.x14,0,0      =  color14,0,0

.d30,70       = delete30-70
.d30-70       = delete30-70
.dc,100       = delete10-100  (Substitui automaticamente "c" pela
                               primeira linha do programa)
.d200,f       = delete200-300 (Substitui automaticamente "f" pela
                               última linha do programa)
.dc,f         = delete10-300

.w20     =  width20

.30      =  list30
.30,50   =  list30-50
.30-50   =  list30-50
```

23.4 - COMANDOS BIT-BASIC X PROGRAMAS SIMPLIFICADOS (FUNÇÕES).

.hb000,b020
São mostrados na Tela todos os BYTES entre #b000 e #b020, na representação "Hexadecimal", já preparando um comando ".p" para o caso de se desejar modificar o conteúdo destes BYTES.

.pb000,cd c0 00 c9
Os BYTES que aparecem "Após a Vírgula" (representação Hexadecimal, dois caracteres para cada BYTE, separados por um caráter "Branco" ou outro qualquer), são colocados na memória (POKE'S) a partir do endereço #B000.

.hx
Aguarda o pressionamento de qualquer tecla que corresponda a um caráter ASCII (inclusive teclas "Especiais" tais como "ESC", "TAB" e outras), e mostra na Tela o caráter na representação ASCII, Hexadecimal e Binária.

.fea,prog1
Mostra os endereços de Início, Fim e Execução para o programa PROG1 (programa executável, em linguagem de máquina, salvo por "Bsave").

.opa,pgtex,12
Executa o "Open" do arquivo pgtex e efetua a leitura dos 12 X 100 = 1200 primeiros BYTES.

.fd
Lista na Tela, no formato ASCII, o conteúdo do arquivo "Aberto" pelo comando ".opa", a partir do BYTE posicionado naquele comando (no caso do exemplo, lista o arquivo pgtex a partir do caráter 1201).
A cada "tela cheia" aguarda o pressionamento de qualquer tecla para continuar o processo (inclusive antes da primeira tela).

.fx
Lista na Tela, nos formatos ASCII e Hexadecimal, o conteúdo do arquivo "Aberto" pelo comando ".opa", a partir do caráter posicionado (semelhante comando ".fd").

.q1
Executa o comando "close#1" para "fechar" o arquivo aberto por um comando ".opa".

..
Aciona o comando ".wwf", que é transformado no comando BIT-BASIC "<<xxx", onde "xxx" é o número da última linha do programa BASIC existente na memória (lista o programa a partir de sua última linha).

.ok
Faz com que não seja mais apresentada a mensagem "Ok" após a execução de uma linha de comando BASIC no modo "Direto".
Repetindo o comando a mensagem volta a ser apresentada.

23.5 - CARACTERES DE CONTROLE.

A rotina SINTAXE monta o comando em uma "Área de Trabalho" antes de passá-lo ao BASIC para execução.
A TABELA DE SINTAXE contém "Caracteres de Controle" intermeados a valores constantes, que dirigem a rotina nesta tarefa.
São os seguintes os "Caracteres de Controle" e os seus significados correspondentes:

#FF - Neste ponto deve ser obtido o próximo parâmetro do Comando BIT-BASIC e colocado na Área de Trabalho (os parâmetros são separados por vírgula = #2C).

#FC - Se não há mais nenhum parâmetro fornecido, encerra a conversão, passando o comando ao BASIC na situação em que se encontra. Caso contrário, continua o processo de conversão.
(Para o caso de "Parâmetros Opcionais".)

#F8 - Coloca um caráter obtido da linha de comando BIT-BASIC na Área de Trabalho, e "Salva" este caráter para utilização posterior.

#F9 - Coloca na Tela o caráter previamente "Salvo" quando o último caráter de controle #F8 foi processado.

#FE - Funciona de forma semelhante ao caráter #FF, porém antes é feita verificação se o parâmetro obtido é "c" (#63) ou "f" (#66), substituindo-os pelo "Número da Primeira Linha BASIC" ou pelo "Número da Última Linha BASIC" respectivamente.

#FB - Indica que a linha de comando deve ser apresentada na Tela, antes de ser executada pelo BASIC.

#00 - Indica o fim de uma entrada na TABELA, ponto no qual a linha de comando BASIC já está completa.

#FA - Indica o "Fim da TABELA", quando o comando BIT-BASIC não é nela encontrado.

Se o caráter obtido da TABELA não é nenhum dos anteriores, o procedimento da rotina SINTAXE é colocá-lo na Tela e passar para o caráter seguinte.

23.6 - MODIFICAÇÃO DA TABELA DE SINTAXE.

O programa BITBAS.TAB que acompanha o BIT-BASIC serve para modificar a TABELA DE SINTAXE.

As primeiras linhas deste programa são linhas de "Comentário" (precedidas do caráter "'") e cada uma delas corresponde a uma Entrada na Tabela.
As duas primeiras posições de cada linha, após o "'" correspondem à "Identificação do Comando" (a serem utilizadas após o "." no comando BIT-BASIC).
Se a segunda posição for "Branco", somente um caráter é necessário para identificar o comando.

Os caracteres de controle devem ser colocados entre os símbolos "[]" e serão convertidos para os caracteres já descritos no item 23.3, sendo colocados em uma "Área de Trabalho".
Os caracteres de controle válidos são os seguintes.

[+] - Obtém próximo parâmetro do comando BIT-BASIC e coloca na Área de Trabalho.
[-] - Passa o comando ao Interpretador BASIC caso não haja mais parâmetros fornecidos.
[<] - Obtém um caráter da linha de comando BIT-BASIC, coloca-o na Área de Trabalho e o "Salva" para uso posterior.
[>] - Obtém último caráter salvo em [<] e o coloca na Área de Trabalho.
[=] - Se próximo parâmetro é um caráter "c" ou "f", o substitui pelo número da primeira/última linha do programa BASIC presente na memória.
[$] - A linha de comando deve ser apresentada na Tela antes de ser processada pelo BASIC.

Ao ser executado o programa BITBAS.TAB, este efetua o carregamento do BIT-BASIC e "modifica" a TABELA DE SINTAXE de acordo com os critérios acima descritos.
O BIT-BASIC é então novamente "salvo" sob outro nome, com a TABELA já modificada, para evitar que a versão original possa ser danificada em razão de um erro acidental.

23.7 - LISTAGEM DO PROGRAMA BITBAS.TAB

```
0 'l  load"[+]:[+]"
20 'r  run"[+]:[+]"
30 'sO screenO
40 'sl screenl
50 'sy _system
60 's  save"[+]:[+]"
70 'ko keyon
80 'kf keyoff
90 'k  kill"[+]:[+]"
100 'bl bload"[+]:[+]"[-],[+]
110 'bs bsave"[+]:[+]",&h[+],&h[+][-],&h[+]
120 'v  renum[+][-],[+][-],[+]
130 'd  delete[=][-]-[=]
140 'x  color[+][-],[+][-],[+]
150 'u  defusr=&h[+]:a=usr(0)
160 'n  name"[>][<]:[+]"as"[<]:[+]"
170 'p  x%=&h[+]:a$="[+]":z%=((len(a$)-1)/3):fori%=0toz%:a%=
val("&h"+mid$(a$,3*i%+1,1)+mid$(a$,3*i%+2,1)):poke(x%+i%),a%
:nexti%:print".h";hex$(x%);",";hex$(x%+z%);:end
180 'hx a$=input$(1):a%=asc(a$):print"c=";a$;" h=";hex$(a%);
" d=";a%;" b=";right$("00000000"+bin$(a%),8)
190 'h  a%=&h[+]:print".p";hex$(a%);",";:fori%=a%to&h[+]:a%=
peek(i%):printhex$(int(a%/16));hex$(a%mod16);" ";:nexti%
200 'fe open"[+]:[+]"forinputas#1:r$=input$(7,1):close#1:pri
nthex$(256*asc(mid$(r$,3,1))+asc(mid$(r$,2,1)));",";:printhe
x$(256*asc(mid$(r$,5,1))+asc(mid$(r$,4,1)));",";:printhex$(2
56*asc(mid$(r$,7,1))+asc(mid$(r$,6,1)))
210 'op open"[+]:[+]"forinputas#1[-]:k%=[+]:fori%=1tok%:a$=i
nput$(100,#1):nexti%
220 'fd forx%=0to9999:b$=input$(1):cls:print"  ";k%:fory%=0t
o9999:t=asc(a$):a$=input$(1,#1):k%=k%+1:printchr$(t);:p%=csr
lin:ifp%>21thennextx%:elsenexty%
230 'fx forx=0to9999:b$=input$(1):cls:forr=1to20:print:print
k%;:K%=K%+8:locate6:forj=0to7:t=asc(a$):printhex$(int(t/16))
;hex$(tmod16);" ";:a$=input$(1,#1):p=pos(0):locatep/3+27:ift
<32thenprint".";:locatep:nextj,r,x:elseprintchr$(t);:locatep
:nextj,r,x
240 'f  files[-]"[+]:[+]"
250 'q  close#[+]
260 '.  [$].wwf
270 'ww [$]<<[=]
280 'w  width[+]
290 'ok a%=peek(&hff07):poke&hff0a,&h41:poke&hff09,&h34:poke
&hff08,&hc3:ifa%=&hc9thenpoke&hff07,&hc1:elsepoke&hff07,&hc9
300 '[] fim da tabela
310 '
```

```
320 FORI%=&HC100TO&HC197:READA$:X%=VAL("&H"+A$):POKEI%,X%:NE
XTI%
330 BLOAD"A:BITBAS.ASS"
340 DEFUSR=&HC100:A=USR(0)
350 BSAVE"A:BITBAS1.ASS",&HA500,&HC0FF,&HC020
360 DEFUSR=&HC020:A=USR(0):END
370 DATA21,00,a5,11,ff,b4,DD,2A,76,F6
380 DATADD,7E,07,FE,5B,CA,94,C1,CD,8E
390 DATAC1,DD,7E,08,CD,8E,C1,DD,E5,DD
400 DATA7e,0A,FE,00,28,0B
410 DATAFE,5B,28,16,CD,8E,C1,DD,23,18
420 DATAEE,CD,8E,C1,DD,E1,DD,22,3A,C1
430 DATADD,2A,00,00,18,CC,DD,23,DD,7E
440 DATA0A,FE,2B,20,0B,3E,FF,CD,8E,C1
450 DATADD,23,DD,23,18,CB,FE,2D,20,04
460 DATA3E,FC,18,EF,FE,3E,20,04,3E,F8
470 DATA18,E7,FE,3C,20,04,3E,F9,18,DF
480 DATAFE,3D,20,04,3E,FE,18,D7,FE,24
490 DATA20,04,3E,FB,18,CF,FE,26,20,0C
500 DATADD,E1,DD,22,86,C1,DD,2A,00,00
510 DATA18,93,DD,23,18,8F,77,23,E7,D8
520 DATAC1,C1,3E,FA,77,C9
```

CAPÍTULO III

O PROCESSADOR Z-80

01 - O PROCESSADOR Z-80

O Z-80 É O PROCESSADOR PRINCIPAL DOS MICROCOMPUTADORES MSX.

Para explicar o seu funcionamento vamos compará-lo ao BASIC, com o qual você já está familiarizado.
Quando você escreve um "Programa BASIC", e comandada a sua execução (RUN), o "Interpretador BASIC" (instalado na ROM) percorre linha a linha as "Instruções" deste programa por você criadas, efetuando uma a uma as operações solicitadas (INPUT, PRINT, etc).
Neste sentido o Z-80 funciona de forma semelhante ao Interpretador BASIC, pois também percorre uma sequência de "Instruções" criadas pelo usuário, executando uma a uma as operações determinadas.

CARACTERÍSTICAS Z-80 X "INTERPRETADOR BASIC"

Embora semelhantes no modo de operação, algumas diferenças e caracteristicas importantes dos dois "Programas" devem ser evidenciadas:

- O BASIC fica instalado em memória ROM, enquanto que o Z-80 é constituido por uma "Microplaqueta" ("CHIP") que armazena um circuito eletrônico muito mais complexo (do nosso ponto de vista, porém, esta diferença não será considerada).

- A "Linguagem" que o Interpretador BASIC entende é o "BASIC", enquanto que a "Linguagem" que o Z-80 entende é a "Linguagem de Máquina do Z-80" (abordaremos esta linguagem, o ASSEMBLER Z-80, mais de perto).
  Escrever um Programa BASIC (Linguagem BASIC) para ser executado pelo Interpretador BASIC é equivalente a excrever um programa Z-80 (Linguagem ASSEMBLER Z-80) para ser executado pelo processador Z-80).

- No BASIC você pode definir vários "Campos" ou "Áreas de Trabalho" (por exemplo "A$", "I", "B%(I), etc.), e existem outras "Áreas de Trabalho" do próprio BASIC que podem ser acessadas (por exemplo "KEY1", "KEY2", "SPRITE$(0)", etc.).
  No Z-80 também existem áreas "Pré-Definidas", constituídas pelos "Registradores", "Pilha do Sistema" e outras, cujo funcionamento será posteriormente abordado que podem ser também definidos "Campos de Trabalho" do Usuário (uma caracteristica das Instruções do Z-80 é que suas áreas pré-definidas são "Obrigatoriamente" referenciadas na maioria delas).

- A linguagem BASIC é de "Alto Nível" (flexível e próxima da linguagem "Humana"), enquanto que a linguagem Z-80 é de "Baixo Nível" (mais "distante" da linguagem "Humana", apenas operações elementares estritamente vinculadas ao Z-80).

- O programa BASIC segue a ordem dos "Números" das Linhas" e, dentro destas, sequencialmente, as instruções ali contidas (a não ser que um "Desvio" seja provocado por uma Instrução GOTO/GOSUB).
  O Z=80 segue a "Sequência Física" das Instruções, armazenadas umas imediatamente após as outras (a não ser que um "Desvio" seja provocado).

- O próprio "Interpretador BASIC" é "Escrito" na "Linguagem de Máquina do Z-80".
  Portanto, o Z-80 "Executa" o "Interpretador BASIC" que, por sua vez, "Controla a Execução" do "Programa BASIC" escrito por você (utilizando o próprio Z-80 para isto).

- Além da tarefa de "Controlar a Execução do Programas BASIC do Usuário", o "Interpretador BASIC" executa todo o trabalho de "Comunicação com o Usuário MSX", ou seja, é ele que analisa qualquer comando por você escrito, acionando todas as rotinas apropriadas para o seu tratamento.

ÁREAS DE TRABALHO E LINGUAGEM DO Z-80

Passaremos a seguir a descrever as "Áreas de Trabalho" do Z-80 e a sua "Linguagem".
Pretendemos ser extremamente objetivos e práticos nas explicações, com as quais você poderá "Iniciar" uma comunicação "Direta" com o Z-80, até então somente efetuada com a "Intermediação" do Interpretador BASIC.

02 - ÁREAS DE TRABALHO DO Z-80

São as seguintes as principais "Áreas de Trabalho" do "Programa Z-80" que interessam para os nossos objetivos.

REGISTRADORES B, C, D, E, H e L
06 "Áreas de Trabalho" de 01 BYTE cada uma.
Estas áreas são denominadas REGISTRADORES, e podem também ser utilizadas "aos Pares" (BC,DE,HL).

REGISTRADORES IX, IY
02 "Áreas de Trabalho" de 02 BYTES cada uma.
Estas áreas são utilizadas como "Índice" por algumas Instruções do Z-80.

REGISTRADOR A
Uma área de trabalho "Especial", de um BYTE.

REGISTRADOR F
Uma área de trabalho "Especial", de um BYTE.

PILHA DO SISTEMA
Área de trabalho de tamanho variável, destinada a "Salvar" o conteúdo das outras áreas de trabalho.

REGISTRADORES SP e PC
Duas áreas de trabalho "Especiais", de dois BYTES cada uma.

O QUE FAZ O PROCESSADOR Z-80

O que a maioria das "Instruções" do Z-80 fazem é "Movimentar" dados entre as suas Áreas de Trabalho, e entre elas e as memórias ROM/RAM.
Portanto, "Escrever um Programa Z-80" significa, basicamente, escrever uma série de instruções para colocar dados, retirar dados, modificar dados ou testar os dados contidos nestes locais.

MUITO SIMPLES, NÃO É ?

DESCRIÇÃO DAS ÁREAS DE TRABALHO DO Z-80

02.1 - REGISTRADOR A

O Registrador "A" (Accumulator = Acumulador) é "Especial" porque um grande número de Instruções do Z-80 trabalham exclusivamente com esta área de trabalho.

02.2 - REGISTRADOR F

O Registrador "F" é "Especial" porque é utilizado pelo Z-80 para Registrar/Indicar "O QUE ACONTECEU" ao ser executada uma Instrução determinada.

Cada BIT do Registrador F guarda uma informação diferente e, assim, uma determinada Instrução Z-80 pode "Posicionar" (Ligar ou Desligar) cada um dos OITO BITS deste Registrador (também chamados de "Indicadores de Estado").
Por exemplo, se o Z-80 executa uma instrução de "Subtração" de um certo valor do Registrador A, o BIT-6 do Registrador F ficará "Ligado/Desligado" para indicar se o resultado da operação é "Igual" ou "Diferente" de Zero.

Estes "Indicadores" também podem ser "Ligados/Desligados" por instruções do Z-80 próprias para esta finalidade, ou podem ter seu posicionamento "Forçado" pelo programa do usuário, com o objetivo de registrar "o que aconteceu" durante a execução de uma determinada rotina, de acordo com critério estabelecido pelo próprio usuário.

São os seguintes os significados dos BITS do Registrador F, que refletem as condições de execução de outras instruções :

S	Z		AC		P/O	N	CY
BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0

BITS DO REGISTRADOR F (INDICADORES DE ESTADO)

- BIT-0 (CY) Indicador de Transporte (CY = CarrY-status)
- BIT-1 (N) Indicador de Adição/Subtração
- BIT-2 (P/O) Indicador de Paridade/Estouro (P/O = Parity/Overflow)
- BIT-4 (AC) Indicador de Transporte Intermediário (AC = Auxiliary Carry)
- BIT-6 (Z) Indicador de "Zero" (Z = Zero)
- BIT-7 (S) Indicador de Sinal Algébrico (S = Sign)

(O BIT-0 é o BIT "Mais à Direita" ou "BIT de mais Baixa Ordem", e o BIT-7 é o BIT "Mais à Esquerda" ou "BIT de mais Alta Ordem").
Destes, somente os indicadores CY e Z interessam mais de perto aos nossos propósitos.

INDICADOR DE TRANSPORTE (CARRY-STATUS - CY)
O Indicador CY é "Ligado" pelo Z-80 (valor 1) quando uma instrução "Desloca para Fora" (Transporta") o BIT mais à esquerda do campo onde é colocado o resultado da operação. Por exemplo, quando o resultado da Soma de um valor ao Registrador A "excede" a sua capacidade (255 = #FF), ou quando uma subtração resulta "Negativa".

INDICADOR DE ZERO (Z)
O Indicador Z é "Ligado" pelo Z-80 (Valor 1) quando uma operação resulta em valor Zero, e "Desligado" (Valor 0) quando o resultado é diferente de Zero.
Por exemplo, após a instrução de "Subtração do Registrador A Dele Próprio" o resultado é Zero, e Z estará "Ligado".

CÓDIGOS DE "LIGADO/DESLIGADO" PARA CY E Z
CY = 1 (C) Ligado (ocorreu Transporte)
CY = 0 (NC) Desligado (não ocorreu Transporte)
Z = 1 (Z) Ligado (resultado Zero)
Z = 0 (NZ) Desligado (resultado diferente de Zero)

02.3 - PILHA DO SISTEMA
É uma área de trabalho que inicia em um endereço da memória RAM, que pode ser escolhido pelo usuário, e é constituída por um "Espaço de Endereços Reservados", onde podem ser "Empilhados" vários "Blocos de DOIS BYTES cada um".
Existem "Instruções do Z-80" próprias para "Colocar" e "Retirar" estes "Pacotes" de dois BYTES na/da PILHA.
A PILHA DO SISTEMA existe para "Salvarmos" temporariamente, de uma maneira simples e rápida, o conteúdo das Áreas de Trabalho do Z-80, enquanto elas são utilizadas com outros valores por outras rotinas, recuperando posteriormente os valores originais.
É importante informar que nunca podem ser "Esquecidos" pacotes na pilha, pois ela compartilha a memória RAM com os programas, e poderá "destruir" estes programas se crescer além da área previamente reservada (o Z-80 não controla estes limites).

02.4 - REGISTRADOR SP (APONTADOR DE PILHA - STACK-POINTER)
O Registrador SP (STACK-POINTER) é outra "Área de Trabalho" do Z-80, sendo um campo de dois BYTES que indica para o Z-80 o endereço do "Bloco Atual Disponível" na PILHA DO SISTEMA.
É interessante observar que a PILHA "cresce para Baixo", ou seja, os pacotes são empilhados no sentido do maior para o menor endereço na memória.
É interessante também observar que o Indicador SP aponta sempre para uma posição "Livre", imediatamente seguinte à posição "Ocupada" pelo último pacote armazenado na PILHA.

02.5 - REGISTRADOR PC (CONTADOR DE PROGRAMA - PROGRAM-COUNTER)
A última área de trabalho do Z-80 aqui citada é o Registrador PC (PROGRAM-COUNTER), que é um campo de dois BYTES onde o Z-80 guarda o endereço da "Instrução Seguinte" àquela que está sendo executada.

03 - LINGUAGEM DO Z-80

Como já dissemos, a maioria das operações que comandamos ao Z-80 para execução estão relacionadas às suas áreas de trabalho e à memória RAM/ROM, e se destinam a tratar com o seu conteúdo.

As "Tarefas" (ou "Funções") que o Z-80 pode executar são pré-definidas, assim como a "Linguagem" que comanda estas tarefas, e são elas que o caracterizam e diferenciam dos demais processadores existentes (da mesma forma, as Funções e Sintaxe do BASIC é que o diferenciam das demais linguagens).

Cada Função a ser executada possui um "Código" ou "Palavra" associada, que pode ocupar um ou dois BYTES, e corresponde a uma operação específica a ser executada sobre uma das áreas citadas.
Assim, exitem "Palavras" para "Movimentar BYTES" de um lugar para outro, "Palavras" para "Somar o conteúdo de duas áreas de trabalho", etc.
Estas palavras, mais a "Sintaxe" empregada para a sua utilização (regras de escrita), constituem a "LINGUAGEM DO Z-80".

Vejamos um exemplo de uma "Palavra" pertencente à "Linguagem do Z-80" e que pode ser por ele entendida e processada.
Suponhamos que os BYTES que serão percorridos para buscar e executar as "Instruções do Programa" contém os seguintes valores (Representação Hexadecimal) :

#C5 #3E #05 #87 . . .

Ao analisar o primeiro BYTE do programa (#C5), o Z-80 "Reconhece" o seu conteúdo como uma "Palavra" (ou Instrução) que solicita a execução da função "Armazene o conteúdo do Par de Registradores BC na PILHA DO SISTEMA".
Como nenhuma informação adicional é necessária para a execução desta função, o Z-80 executa imediatamente esta ordem, colocando o conteúdo do REG-BC na PILHA e atualizando o Registrador SP, e passa para a próxima "Palavra" que, neste caso, começa no BYTE imediatamente seguinte, pois a presente Instrução ocupa apenas UM BYTE (é o PC - CONTADOR DE PROGRAMA, que guarda este endereço, lembra-se ?).

Ao examinar o BYTE seguinte (#3E), ele "Reconhece" a solicitação da Tarefa "Coloque no Registrador A o valor contido no BYTE imediatamente seguinte" ao BYTE #3E".
Prontamente o Z-80 "Obedece" e coloca no Registrador A o valor #05.

Neste caso, a "Instrução Z-80" "ocupou" dois BYTES, o primeiro indicando "O QUE FAZER" (#3E), e o segundo fornecendo um PARÂMETRO (#05 - no caso uma constante) para ser utilizado na sua execução.
Portanto, o "Tamanho" de cada Instrução é variável, dependendo do Tipo de Função a ser executada (o Z-80 "Conhece" os tamanhos de cada uma de suas Instruções).

O BYTE seguinte à Instrução Z-80 "#3E #05" contém o valor #87, e informa ao Z-80 que ele deve "somar o valor contido no Registrador A ao próprio Registrador A".
Como a Instrução anterior colocou em REG-A o valor #05, após esta Instrução o Registrador A conterá #0A, que é o resultado da soma (#05 + #05).
Além disto, o Indicador CY estará "Desligado - NC", pois não ocorreu "Estouro", e o Indicador Z estará "Desligado - NZ", pois o resultado "#0A" é "Diferente de Zero".

A LINGUAGEM ASSEMBLER

Como você deve ter percebido, a "Linguagem de Máquina" do Z-80 não é "Fácil" de ser memorizada e utilizada, pois somente "Reconhece" símbolos representados por combinações de valores compreendidos entre #00 e #FF (Menor e Maior valores que podem estar contidos em um BYTE).

Para contornar este problema, visando tornar mais "Fácil" a elaboração de programas para o Z-80, foi construída uma "Outra Linguagem", utilizando símbolos de tratamento mais práticos e de fácil memorização pelas pessoas.
Esta "Linguagem" é muito "Próxima" da Linguagem de Máquina do Z-80 (cada "Palavra" desta linguagem tem correspondência quase "Direta" com as "Palavras" reconhecidas pelo Z-80), e por este motivo também é considerada uma linguagem de "Baixo Nível".

O NOME DESTA LINGUAGEM É "ASSEMBLER DO Z-80".

Obviamente, é necessário "Um Novo Programa" para "Transformar" (ou "Montar", ou ainda "Compilar") um programa escrito em "ASSEMBLER Z-80" para a "Linguagem de Máquina do Z-80" (o Código em Linguagem de Máquina assim gerado é denominado "Código Objeto").

O NOME DESTE PROGRAMA É "COMPILADOR ASSEMBLER DO Z-80".

É interessante observar que, como cada Microprocessador "Fala" uma Linguagem própria e exclusiva, cada um deles também têm o seu ASSEMBLER (MONTADOR) próprio.

LINGUAGEM ASSEMBLER - EXEMPLO

As "Instruções ASSEMBLER" correspondentes às Instruções do Z-80 utilizadas no último exemplo analisado seriam escritas da seguinte maneira :

```
0010 INICIO:      PUSH BC
0020              LD   A,#03
0030              ADD  A,A
```

0010 (PUSH = "Empurre") Comanda ao Z-80 a Função "Armazene na PILHA DO SISTEMA" o valor contido no Registrador BC.
0020 (LD = LOAD = "Carregue") Comanda ao Z-80 a função "Carregue" ou "Armazene" o valor #03 no Registrador A.
0030 (ADD = "Some") Comanda ao Z-80 a fução "Some" o valor contido no Registrador A ao próprio Registrador A.

O "Programa COMPILADOR ASSEMBLER" "lê" os comandos assim escritos e os transforma na "Linguagem de Máquina Z-80", que é entendida por este processador (#C5 #3E #05 #87).

MUITO MAIS FÁCIL, NÃO É MESMO ?

A "Sintaxe" (regras de codificação) exigida pelo ASSEMBLER pode variar de compilador para compilador.
Para o compilador utilizado para o BIT-BASIC as regras são as seguintes :

Exemplo : 0010 R01:   LD   A,#20

NÚMERO : 0010
Cada linha do "Programa ASSEMBLER" é numerada, à semelhança da Linha de Programa BASIC (Primeiras posições da Instrução).

NOME : R01
Além do número, uma instrução "Pode Ter" (opcional) um "Nome" colocado em seguida ao "Número", que poderá ser utilizado por outras instruções, onde representará o "Endereço" desta instrução.

FUNÇÃO : LD
Em seguida deve ser colocada a "Função" a ser executada por esta Instrução ASSEMBLER (Movimentação, Soma, etc.).

PARÂMETROS : A,#20
Logo após devem aparecer os "Parâmetros" sobre os quais a Função será executada (Áreas de Trabalho do Z-80, Áreas de Trabalho do seu próprio programa na memória ROM/RAM, Valores Constantes, etc.).

OBSERVAÇÃO : Quando existem dois parâmetros a operação é normalmente efetuada do Segundo para o Primeiro (no exemplo acima a constante #20 é carregada em REG-A).

04 - INSTRUÇÕES ASSEMBLER

Neste item mostraremos as "Palavras" da linguagem ASSEMBLER que determinam ao Z-80 as funções a serem executadas, com a descrição destas funções.
Os "Códigos Z-80" correspondentes serão também mostrados (entre parêntesis).

Inicialmente descreveremos cada "Tipo de Função" e em seguida as "Variações" que ela pode sofrer, dependendo do "Tipo de Parâmetro" com o qual está trabalhando (um Registrador, um Endereço, uma Constante, etc.).
Os diferentes tipos serão apresentados com a utilização de exemplos.

A LISTA COMPLETA DAS INSTRUÇÕES ASSEMBLER ESTA NO APÊNDICE 07.

Para nos referirmos aos Registradores do Z-80 utilizaremos sempre, por simplificação, o prefixo "REG-", seguido pela identificação do Registrador ou Par de Registradores.

ADD

ADD = "SOME"

Estas instruções ASSEMBLER correspondem a Instruções Z-80 cuja função é somar o, valor de uma Área de Trabalho ou o valor contido em um endereço de memória, em outra Área de Trabalho do Z-80.
A Instrução BASIC X%=X%+Y% executa função semelhante.

Os Indicadores de Estado são posicionados de acordo com o resultado da soma.
Por exemplo, se ocorrer um "Estouro" (o resultado não "Cabe" na Área onde deve ser colocado), o Indicador de Transporte CY é "Ligado"

```
0010          ADD  A,(HL)     (#86)
0020          ADD  A,#A8      (#C6 #A8)
0030          ADD  HL,HL      (#29)
```

0010 Soma em REG-A o valor do BYTE apontado por REG-HL.
0020 Soma o valor #A8 em REG-A.
0030 Soma REG-HL em REG-HL ("Dobra" o valor contido em REG-HL).

INDICADORES DE ESTADO CY,Z
(Primeiro Operando = Par de Registradores)
(Instrução 0030 do Exemplo)

Não são afetados.

INDICADORES DE ESTADO CY,Z
(Primeiro Operando = Registrador Único ou Endereço de Memória)
(Instruções 0010 e 0020 do Exemplo)

```
Resultado menor do que Zero ou "Estouro" : C,NZ
Resultado igual a Zero                   : NC,Z
Resultado maior do que Zero              : NC,NZ
```

AND/OR/XOR

AND = "E"
OR = "OU"
XOR = EXCLUSIVE OR = "OU EXCLUSIVO"

Estas Instruções ASSEMBLER dão origem a Instruções Z-80 que efetuam a "COMPARAÇÃO BIT A BIT" do conteúdo do Acumulador (REG-A) com o conteúdo de outro Registrador ou posição de memória.
O resultado da comparação de cada par de BITS é recolocado na posição correspondente de REG-A.
Os Indicadores de Estado são posicionados para refletir as condições da comparação.

As Instruções BASIC "A=AORB", "X=XANDY, Z=ZXORA", por exemplo, executam funções semelhantes sobre as áreas de trabalho especificadas (também BIT a BIT).

A comparação pode ser feita segundo três critérios diferentes (AND, OR ou XOR), a seguir descritos.

- Modalidade AND (E)
  O BIT resultante é 1 (Ligado) somente se os dois BITS comparados são 1.

REG-A	1 0 1 0 0 0 1 1
BYTE COMPARADO	1 1 0 0 1 0 0 1
RESULTADO	1 0 0 0 0 0 0 1

- Modalidade OR (OU)
  O BIT resultante é 1 (Ligado) se qualquer dos dois BITS comparados é 1.

REG-A	1 0 1 0 0 0 1 1
BYTE COMPARADO	1 1 0 0 1 0 0 1
RESULTADO	1 1 1 0 1 0 1 1

- Modalidade XOR (OU EXCLUSIVO)
  O BIT resultante é 1 (Ligado) somente se apenas um dos dois BITS comparados é 1.

REG-A	1 0 1 0 0 0 1 1
BYTE COMPARADO	1 1 0 0 1 0 0 1
RESULTADO	0 1 1 0 1 0 1 0

```
0010          AND  E        (#A3)
0020          OR   (HL)     (#B6)
0030          XOR  #FF      (#EE #FF)
0040          AND  A        (#A7)
```

0010 Compara REG-A com REG-E, BIT a BIT, na modalidade AND, colocando o resultado em REG-A.

0020 Compara REG-A com o conteúdo do BYTE apontado por REG-HL, BIT a BIT, na modalidade OR, colocando o resultado da comparação em REG-A.

0030 Compara REG-A com o valor #FF, BIT a BIT, na Modalidade XOR, colocando o resultado em REG-A.
É interessante observar que, se REG-A contém inicialmente #00, a cada nova passagem pela instrução "XOR #FF" REG-A conterá, alternadamente, os valores #FF e #00.
Esta é uma técnica interessante para utilizarmos como uma função que "Liga/Desliga" um Indicador em um programa ASSEMBLER.

0040 Compara REG-A com o próprio REG-A, BIT a BIT, na Modalidade AND, colocando o resultado da comparação em REG-A.
É interessante observar que neste caso o valor resultante é sempre #00, e os Indicadores CY e Z resultam sempre (Z,NC).
Podemos utilizar esta instrução para "Zerar" o conteúdo de REG-A ou para "Forçar" o posicionamento dos Indicadores de Estado.

INDICADORES DE ESTADO CY,Z

Resultado igual a Zero : Z,NC
Resultado diferente de Zero : NZ,NC
(Estas Instruções sempre "Desligam" o Indicador CY.)

CALL

CALL = "CHAME"

Estas Instruções ASSEMBLER são Transformadas em Instruções de Máquina do Z-80 que determinam a execução de um certo "Conjunto de Instruções", localizadas em algum endereço na memória, com RETORNO posterior para a instrução imediatamente seguinte à instrução CALL.

Você pode comparar a função desempenhada por esta Instrução Z-80 com a função da Instrução GOSUB do BASIC (que possui a Instrução RETURN para comandar o Retorno ao ponto de chamada).

O Endereço de início da rotina a ser "Chamada" é colocado no "Operando de Endereço" da instrução CALL.

A instrução CALL usa a PILHA DO SISTEMA para "Salvar" o endereço de RETORNO (endereço da Instrução imediatamente seguinte à Instrução CALL).

Para comandar o Retorno no ponto desejado da rotina chamada existe outra Instrução do Z-80 que deve ser utilizada: Instrução RET. (Descrita adiante.)

A chamada também pode ser feita "Sob Condição Determinada".

```
0010          CALL #C000        (#CD #00 #C0)
0020          CALL DESVIO       (#CD #00 #C0)
0030          CALL NZ,#C000     (#FC #00 #C0)
```

0010 Efetua "Chamada" à "Rotina" que inicia no endereço #C000.
0020 Efetua "Chamada" à "Rotina" DESVIO (que inicia no endereço #C000).
0030 Efetua "Chamada" à "Rotina" que inicia no endereço #C000, mas somente se o Indicador Z estiver "Desligado". Caso contrário executa a instrução imediatamente seguinte à instrução CALL.

INDICADORES DE ESTADO CY,Z

Não são afetados.

CCF

CCF = COMPLEMENT CARRY-FLAG =
"COMPLEMENTE INDICADOR-DE-TRANSPORTE"

Esta Instrução ASSEMBLER é compilada em Instrução Z-80 cuja função é "Complementar o valor do Indicador de Estado CY" (CARRY-FLAG - INDICADOR-DE-TRANSPORTE).
O valor do BIT-0 do Registrador F, que desempenha o papel de "Indicador CY", pode conter os valores "0" ou "1".
"Complementar" o valor atual de CY significa "Somar 1" a este valor, com o que o seu conteúdo é "Invertido" pela instrução CCF, ou seja, se for "0" passará a ser "1", e vice-versa.

```
0010            AND  A            (#A7)
0020            CP   #01          (#FE #01)
0030            CCF               (#3F)
0040            JP   C,#B500       (#DA #00 #B5)
```

0010 Coloca #00 no REG-A.
0020 Compara REG-A com #01.
Como o conteúdo de REG-A (#00) é "Menor do que #01", o Indicador CY será "Ligado" ("1" = "C") por esta Instrução.
0030 "Complementa" (ou "Inverte") o valor do Indicador CY. Como CY foi posicionado em "1" pela instrução anterior, após esta instrução seu valor passa a ser "0" ("Desligado" = "NC").
0040 Não desvia para a o endereço #B500, pois a condiçao "C" (CY Ligado) não será satisfeita, passando para a instrução seguinte do programa.

INDICADORES DE ESTADO CY,Z

CY = C ou NC, se o seu valor anterior era NC ou C.
(O Indicador Z não é afetado.)

CP

CP = COMPARE = "COMPARE"

Estas instruções servem para "Comparar" o conteúdo de REG-A com outro valor, "Posicionando" (Ligando/Desligando) os "Indicadores de Estado" do Registrador F.
O Resultado da comparação pode ser verificado posteriormemte por outras instruções (Analisando os Indicadores de Estado) e diferentes procedimentos podem ser executados em função deste resultado.

A instrução BASIC "IF A%=&H3B", por exemplo, executa função semelhante.

```
0010          CP   #3B      (#FE #3B)
0020          CP   D        (#BA)
0030          CP   (HL)     (#BE)
```

0010 Compara REG-A com #3B.
0020 Compara REG-A com REG-D.
0030 Compara REG-A com o BYTE que está no endereço apontado pelo REG-HL.

INDICADORES DE ESTADO CY,Z

```
REG-A "Menor" que Operando  : C,NZ
REG-A "Igual" ao Operando   : NC,Z
REG-A "Maior" que Operando  : NC,NZ
```

CPIR

CPIR = COMPARE / INCREMENT REGISTER =
"COMPARE / INCREMENTE REGISTRADOR"

Esta Instrução ASSEMBLER é compilada em Instrução de Máquina do Z-80 que efetua as seguintes operações :

- Compara o conteúdo de REG-A com o BYTE apontado pelo endereço contido em REG-HL.
- Se a comparação resultar "Igual", passa para a instrução imediatamente seguinte à instrução CPIR (neste caso com o Indicador Z "Ligado").
- Subtrai 1 de REG-BC.
- Se o resultado da subtração for "Zero", passa para a instrução seguinte (neste caso , com o Indicador Z "Desligado").
- Se o resultado da subtração for "Diferente de Zero", Soma 1 a REG-HL e repete todas as operações anteriores.

Esta instrução é utilizada para "Procurar" um valor determinado, BYTE a BYTE, a partir de um endereço, em um trecho de memória determinado.
Todas as operações de busca são efetuadas com APENAS UMA Instrução de Máquina, sob controle "Interno" do Z-80.

```
0010          LD   HL,#B000   (#21 #00 #B0)
0020          LD   BC,#0005   (#01 #05 #00)
0030          LD   A,#AA      (#3E #AA)
0040          CPIR            (#ED #B1)
0050          JP   Z,ENCONT   (#CA #D5 #CD)
0060          JP   NAOENCONT  (#C3 #55 #CA)
```

0010 Carrega #B000 em REG-HL.
0020 Carrega #0005 em REG-BC.
0030 Carrega #AA em REG-A.
0040 Compara REG-A com cada BYTE, a partir do endereço #B000 (REG-HL), até encontrar um cujo valor seja "Igual" a #AA (REG-A), ou até terem sido comparados 05 BYTES (REG-BC).
0050 Se um valor igual ao de REG-A (#AA) foi encontrado, desvia para a instruçao ENCONT (Endereço #CDD5).
0060 Se um valor igual ao de REG-A não foi encontrado, desvia para a instrução NAOENCONT (Endereço #CA55).

INDICADORES DE ESTADO CY,Z

Z - Valor procurado foi encontrado.
NZ - Valor procurado nao foi encontrado.
CY - Não é afetado.

As seguintes Instruções BASIC executariam funcão semelhante.

```
FORI=&HB000TO&HB004:A=PEEK(I):IFA<>&HAATHENNEXTI:ELSEGOTOENCONT
```

DEC/INC

DEC = DECREMENT = "DECREMENTE"
INC = INCREMENT = "INCREMENTE"

Estas instruções ASSEMBLER são Compiladas em Instruções Z-80 que efetuam as operações de "Somar 1" ou de "Subtrair 1" dos registradores ou endereços indicados.
A Instrução BASIC X%=X%+1, por exemplo, executa função semelhante.

```
0010          DEC  A        (#3D)
0020          DEC  HL       (#2B)
0030          INC  (HL)     (#34)
0040          INC  SP       (#33)
```

0010 Decrementa 1 (Subtrai 1) de REG-A.
0020 Subtrai 1 de REG-HL.
(O Par de Registradores HL é tratado como se fosse um "Número Único", com o que é possivel trabalhar com valores entre #0000 e #FFFF.)
0030 Incrementa 1 (Soma 1) no BYTE apontado pelo endereço contido em REG-HL.
Se REG-HL contém #D000, por exemplo, o BYTE contido neste endereço terá seu valor acrescido de 1.
0040 Soma 1 ao Registrador SP (Apontador de Pilha - Stack Pointer).

INDICADORES DE ESTADO CY,Z
(Operando = Par de Registradores)
(Instruções 0020/0040 do Exemplo)

Não são afetados.

INDICADORES DE ESTADO CY,Z
(Operando = Registrador Único ou Endereço de Memória)
(Instruções 0010 e 0030 do Exemplo)

Resultado menor do que Zero : NZ
Resultado igual a Zero : Z
Resultado maior do que Zero : NZ
(O Indicador CY não é afetado)

DJNZ

DJNZ = DECREMENT/JUMP-RELATIVE/NZ
(SUBTRAIA/DESVIE-RELATIVO/NZ)

Esta instrução ASSEMBLER corresponde a uma Instrução Z-80 que efetua os seguintes procedimentos :

- DECREMENTA (ou subtrai) de 1 o conteúdo do Registrador B.
- Se o valor resultante em REG-B for "DIFERENTE DE ZERO" (NZ), o Z-80 "Desvia"para uma instrução de deslocamento RELATIVO em relação à instrução DJNZ.
  Este deslocamento é armazenado de forma semelhante às instruções JR, de acordo com o "Número de BYTES" existentes entre a Instrução DJNZ e a Instrução para onde o desvio deve ser efetuado.

```
0010          LD   B,#05    (#06 #05)
0020          LD   A,#00    (#3E #00)
0030 A01:     ADD  A,B      (#80)
0040          DJNZ A01      (#10 #FD)
0050          RET           (#C9)
```

0010 Carrega #05 em REG-B.
0020 Carrega #00 em REG-A.
0030 Soma REG-B em REG-A.
0040 SUBTRAI 1 de REG-B.
Se o resultado for "Diferente de Zero", DESVIA para A01. (#FD representa um "Deslocamento Relativo" de Dois BYTES "Para Trás" em relação ao endereço onde #FD está colocado).
Se o resultado for "Zero", passa para a Instrução seguinte (0050 RET).

INDICADORES DE ESTADO CY,Z

Não são afetados.

O seguinte programa BASIC executa função semelhante à do programa ASSEMBLER exemplificado :

```
10 B%=5:A%=0
20 A%=A%+B%
30 B%=B%-1:IF B%<>0 THEN GOTO 20
40 RETURN
```

EX

EX = EXCHANGE = "TROQUE"

Estas Instruções ASSEMBLER são convertidas para Instruções Z-80 cuja função é efetuar a "Troca de Conteúdo entre Registradores e Memória".

```
0010          EX   DE,HL     (#EB)
0020          EX   (SP),HL   (#E3)
```

0010 Efetua a troca de conteúdos entre os Registradores DE e HL.
0020 Efetua a troca de conteúdos entre REG-HL e os dois BYTES apontados por REG-SP.

INDICADORES DE ESTADO CY,Z

Não são afetados.

IN

IN = INPUT = "ENTRADA"

Estas Instruções ASSEMBLER são transformadas em Instruções Z-80 cuja função é "Carregar em um Registrador um BYTE obtido a partir de uma PORTA-DE-I/O".
Para os nossos propósitos não é necessário saber como esta instrução funciona "Internamente", sendo suficiente sabermos que ela serve para o Processador Z-80 se "Comunicar" com outros Processadores existentes no seu MSX.
Esta comunicação é feita por intermédio de "Ligações" apropriadas existentes no Z-80, chamadas de PORTAS-DE-I/O (I/O = INPUT/OUTPUT = ENTRADA/SAIDA).
O Z-80 pode se comunicar, por exemplo, com o Processador PPI (Programmable Peripheral Interface - Interface Programável de Periféricos) que, entre outras coisas, "Gerencia" o Teclado do MSX, e "Gerencia" os Bancos de Memória ROM/RAM.
Com o uso desta instrução podemos, por exemplo, descobrir qual a configuração atual destes Bancos de Memória.

Função semelhante é executada pela Instrução INP do BASIC.

```
0010          IN   A,(#A8)    (#DB #A8)
0020          LD   C,#A8      (#0E #A8)
0030          IN   B,(C)      (#ED #40)
```

0010 Transfere para o REG-A o valor de um BYTE contido na Porta de I/O identificada como #A8 (esta é a Porta do PPI).
0020 Carrega o valor #A8 em REG-C.
0030 Transfere para REG-B um BYTE da Porta #A8 (equivalente à Instrução 0010).

INDICADORES DE ESTADO CY,Z

Não são afetados.

JP/JR

JP = JUMP = "PULE" ou "SALTE"
JR = JUMP RELATIVE = "PULE RELATIVO"

Estas Instruções ASSEMBLER são Transformadas em Instruções de Máquina, que determinam ao Z-80 que um "Desvio" deve ser efetuado, ou seja, que a próxima Instrução a ser executada não é aquela imediatamente seguinte à atual, como seria seu procedimento normal, mas sim uma outra instrução cujo endereço é determinado pela Instrução JUMP.

Você pode comparar a função desta Instrução com a função da Instrução "GO TO" ("VÁ PARA") do BASIC.

É possível comandar ao Z-80 que efetue o Desvio desejado somente se uma certa condição for satisfeita por um dos Indicadores de Estado.

```
0010 RT1:      JP   #3E4C      (#C3 #4C #3E)
0020           JP   RT2        (#C3 #B4 #C7)
0030           JP   (HL)       (#E9)
0040 RT2:      JP   C,#3E4C    (#DA #4C #3E)
0050           JP   NZ,#3E4C   (#C2 #4C #3E)
0060           JR   $+5        (#18 #03)
0070           JR   NC,$-10    (#30 #F4)
```

0010 Desvia "Incondicionalmente" para o Endereço #3E4C. Portanto, a próxima instrução a ser processada pelo Z-80, após a Instrução RT1, é aquela que inicia no endereço #3E4C.

0020 Desvia "Incondicionalmente" para o Endereço correspondente à Instrução de Nome RT2.
Observe que o "Código de Instrução" da linha 0020 (#C3) é igual ao da linha 0010, e que o campo de Endereço da instrução 0020 é obtido pelo compilador ASSEMBLER, sendo o endereço onde ele próprio colocou a instrução de nome RT2.

0030 Desvia "Incondicionalmente" para o Endereço apontado por REG-HL.
Se REG-HL contém, por exemplo, #AE4F, a próxima instrução a ser executada pelo Z-80 será aquela que está armazenada neste endereço.

0040 Desvia para a instrução que inicia no endereço #3E4C, mas somente se o Indicador de Estado CY está "Ligado" (Condição C no primeiro "Operando" da Instrução).
Esta técnica pode ser utilizada, por exemplo, para efetuar o desvio somente se uma comparação anterior resultou "Negativa".
Se CY está "Desligado" (NC), o Z-80 continua o processamento na instrução imediatamente seguinte à instrucção JUMP.

0050 Desvia para a instrução que inicia no endereço #3E4C, mas somente se o Indicador de Estado Z está "Desligado" (Condição NZ no primeiro "Operando" da Instrução).
Esta técnica pode ser utilizada, por exemplo, para desviar somente se uma comparação anterior resultou "Diferente de Zero".
Se Z está "Ligado" (Z) o Z-80 continua o processamento na instrução imediatamente seguinte à instrução JUMP.

0060 Desvia "Incondicionalmente" para o "Endereço de Início da Instrução Atual + 5" (quinto BYTE após o início da Instrução 0060).
O símbolo "$" indica ao Compilador o "Endereço de inicio da Instrução atual".

0070 Desvia para o "Endereço de Início da Instrução Atual - 10", mas somente se o Indicador de Estado CY estiver "Desligado".

Como você pode observar, as Instruções JR (JUMP RELATIVE) tem três diferenças principais em relação às Instruções JP (JUMP).

. As Instruções JR ocupam um número menor de posições de memória (dois BYTES, contra três BYTES para instruções JP).
. Nas Instruções JR o endereço de Desvio não muda caso o programa seja "Trocado de Lugar" na memória (o "Deslocamento Relativo" permanece o mesmo).
No caso das Instruções JP, o endereço é armazenado de forma "Explícita" e terá que ser modificado para o novo endereço onde ficará a Instrução de destino "Realocada".
. As Instruções JR somente podem desviar para instruções que estejam numa "Distância" que possa ser representada em um único BYTE (-126 a +129).

INDICADORES DE ESTADO CY,Z

Não são afetados.

LD

LD = LOAD = "CARREGUE"

Estas Instruções ASSEMBLER são "Transformadas" em Instruções de Máquina (pelo programa Compilador) que indicam ao Z-80 para efetuar a "Movimentação" de conteúdos entre determinados "Endereços" de memória ROM/RAM e/ou entre as "Áreas de Trabalho" do Z-80.
As Instruções BASIC "A%=3" e "X1%=Y2%", por exemplo, executam função semelhante.

```
0010 A01:    LD   A,#03        (#3E #03)
0020         LD   B,C          (#41)
0030         LD   A,(HL)       (#7E)
0040         LD   A,(#B000)    (#3A #00 #B0)
0050         LD   A,(A01)      (#3A #5F #BD)
0060         LD   HL,#1234     (#21 #34 #12)
0070         LD   HL,(#1234)   (#2A #34 #12)
0080         LD   HL,A01       (#21 #5F #BD)
0090         LD   HL,(A01)     (#2A #5F #BD)
0100         LD   D,(HL)       (#72)
0110         LD   (IX+10),H    (#DD #74 #0A)
```

0010 Carrega a Constante #03 no REG-A.

0020 Carrega (ou Copia) o conteúdo do REG-C no REG-B.

0030 Carrega no REG-A o BYTE apontado pelo endereço contido no REG-HL (atenção para os Parêntesis que envolvem HL).
Por exemplo, se REG-HL contém #B000, e o BYTE existente no endereço #B000 da memória contém #FF, REG-A conterá #FF após a execução desta instrução.

0040 Carrega em REG-A o BYTE contido no endereço #B000.
Se este BYTE contém #FF, REG-A passará a conter #FF.
(Observe a diferença entre a instrução 0010, sem parêntesis, que carrega uma "Constante" (Segundo Operando) em REG-A, e esta instrução, que contém parêntesis, a qual faz referência a "Outro Endereço" de Memória.)
A "Instrução de Máquina" neste caso ocupa três BYTES, sendo o primeiro o "Código da Função" a ser executada e os dois seguintes o "Endereço a Partir do Qual o BYTE deve ser Movido para o Registrador A".

0050 Carrega no REG-A o BYTE apontado pelo endereço correspondente ao "Nome de Instrução" (ou LABEL) "A01".
Supondo que a "Instrução" que tem este nome foi colocada no endereço #BD5F, REG-A conterá o BYTE que existe neste endereço após a execução desta instrução.
O COMPILADOR ASSEMBLER é quem coloca as instruções a partir de endereços determinados e, posteriormente, "Substitui" as referências a "Nomes de Instruções" pelos endereços correspondentes, produzindo então um programa em "Linguagem de Máquina do Z-80".
(Este processo se denomina "Compilação" do programa.)

0060 Carrega em REG-HL a "Constante" (Valor) #1234 (observe que na Linguagem Z-80 os BYTES que contém #12 e #34 são armazenados de forma "Invertida" na memória).

0070 Carrega em REG-HL os BYTES contidos a partir do Endereço #1234 da memória.
Observe que as funções (Resultados das Instruções) das linhas 0060 e 0070 são diferentes.
Para representar esta diferença a "Linguagem ASSEMBLER" não usa Parêntesis na primeira e os usa na segunda, enquanto que a "Linguagem Z-80" usa "Códigos de Instrução" diferentes (#21 e #2A, respectivamente).

0080 Faz "a mesma coisa" que a instrução da linha 0060, exceto que o Compilador converte o nome A01 pelo endereço da instrução correspondente (#BD5F), em tempo de Compilação (observe que os "Códigos de Instrução Z-80" são iguais para ambas as linhas).

0090 Faz "a mesma coisa" que a instrução da linha 0070, valendo as mesmas observações da linha 0080.

0100 Carrega no REG-D o BYTE existente no endereço apontado pelo REG-HL.
Se REG-HL contém, por exemplo, #BD5F e, neste endereço, há um BYTE que contém #CA então REG-D conterá #CA após a execução desta instrução.
Observe que esta Instrução Z-80 tem apenas um BYTE (#72) não necessitando de "Parâmetros".

0110 Carrega o conteúdo de REG-H no Endereço apontado pelo REG-IX acrescido de 10 unidades.
Se REG-IX contém #C000 e REG-H contém #4E, por exemplo, após esta instrução o BYTE do endereço "#C000 + #0A = #C00A" conterá o valor #4E.
Observe que esta função utiliza dois BYTES para ser "Reconhecida" pelo Z-80 (#DD #74) e um BYTE onde é colocado o valor a ser somado/subtraído ao "Registrador IX" (Registrador de Índice).
Os Registradores IX e IY são empregados como "Índices" em Instruções deste tipo.

INDICADORES DE ESTADO CY E Z

Não são Alterados.

LDIR

LDIR = LOAD / INCREMENT REGISTER =
"CARREGUE / INCREMENTE REGISTRADOR"

Esta istrução é Compilada em Instrução de Máquina do Z-80 que efetua as seguintes operações :

- Transfere o conteúdo do BYTE apontado por REG-HL para o endereço apontado por REG-DE.
- Soma 1 em REG-HL e em REG-DE.
- Subtrai 1 de REG-BC.
- Se o resultado da subtração for "Diferente de Zero", repete todas as operações anteriores.
- Se o resultado da subtração for "Zero", passa para a instrução imediatamente seguinte à instrução LDIR.

(Ela é utilizada para movimentar "Blocos" de BYTES de um lugar para outro na memória, sob controle do Z-80, com APENAS UMA "Instrução de Máquina", pois o "Programa Z-80" executa "Internamente" todos os procedimentos e controles necessários).

```
0010          LD   HL,#B000  (#21 #00 #B0)
0020          LD   DE,#C000  (#11 #00 #C0)
0030          LD   BC,#00FF  (#01 #FF #00)
0040          LDIR           (#ED #B0)
```

0010 Carrega o valor #B000 em REG-HL.
0020 Carrega o valor #C000 em REG-DE.
0030 Carrega o valor #00FF em REG-BC.
0040 "Copia" 255 BYTES (#FF), a partir do endereço #B000, para o endereço #C000.

INDICADORES DE ESTADO CY,Z

Não são afetados.

As seguintes Instruções BASIC executariam função semelhante :

```
HL=&HB000:DE=&HC000:FORBC=255TO0STEP-1:A=PEEK(HL+255-BC):
POKE(DE+255-BC),A:NEXTBC
```

OUT

OUT = OUTPUT = "SAÍDA"

Estas Instruções ASSEMBLER são transformadas em Instruções Z-80 cuja função é "Carregar em uma PORTA-DE-I/O o valor contido em um Registrador ou em um BYTE da memória".
Esta é uma função "Inversa" àquela da Instrução IN e as observações lá efetuadas também são válidas para ela.

Função semelhante é executada pela Instrução OUT do BASIC.

```
0010        LD   A,#F0     (#3E #F0)
0020        OUT  (#A8),A   (#D3 #A8)
```

0010 Carrega #F0 em REG-A.
0020 Envia o Valor #F0 para a PPI, via Porta de I/O "#A8".

```
0030        LD   C,#A8     (#0E #A8)
0040        LD   B,#F0     (#06 #F0)
0050        OUT  (C),B     (#ED #41)
```

0030 Carrega #A8 em REG-C.
0040 Carrega #F0 em REG-B.
0050 Envia o Valor #F0 para a PPI, via porta de I/O "#A8".
(Equivalente à Instrução 0010.)

INDICADORES DE ESTADO CY,Z

Não são afetados.

PUSH/POP

PUSH = "EMPURRE"
POP = "SOLTE"

Estas Instruções ASSEMBLER são "Compiladas" em Instruções de Máquina que determinam ao Z-80 a colocação ou retirada de "Pacotes" (DOIS BYTES) na PILHA DO SISTEMA.
A Área de Trabalho SP (STACK-POINTER - APONTADOR-DE-PILHA) é automaticamente atualizada a cada vez que um "Pacote" é colocado ou retirado da PILHA.

```
0010          PUSH BC
0020          PUSH AF
0030          POP  AF
```

0010 Armazena o conteúdo de REG-BC (dois BYTES) na PILHA DO SISTEMA (imediatamente após o último valor já armazenado).
0020 Armazena os conteúdos de REG-A e de REG-F na PILHA DO SISTEMA.
Portanto, podemos "Salvar" na PILHA os Indicadores de Estado contidos em REG-F.
0030 Recupera o último valor armazenado na PILHA DO SISTEMA e o coloca em REG-AF.
Observe que este valor pode ter sido lá colocado a partir de "Outro" Par de Registradores, o que pode ser utilizado para "Trocar" o conteúdo de pares de Registradores.

INDICADORES DE ESTADO (CY,Z)

Não são alterados.

RET

RET = RETURN = "RETORNE"

Estas instruções ASSEMBLER são transformadas em Instruções Z-80 que detereminam um "Desvio" para o endereço armazenado no "Topo" da PILHA DO SISTEMA (último valor aí armazenado).

A denominação de RETORNO para esta função se deve ao fato de que, na maioria das vezes, o endereço de desvio contido na PILHA foi aí colocado por uma instrução CALL, e a instrução RET efetua então o "Retorno ao Ponto de Chamada" (para a Instrução seguinte ao CALL).
Não há, porém, correspondência obrigatória entre uma instrução RET e uma instrução CALL (veja rotina DESVIO do BIT-BASIC, por exemplo).
A função da Instrução RET do Z-80 é semelhante à função da Instrução RETURN do BASIC.

```
0010 B10:       RET          (#C9)
0020            RET  NZ      (#C0)
```

0010 RETORNA (Desvia) para o último endereço armazenado na PILHA DO SISTEMA.
0020 DESVIA para o último endereço armazenado na PILHA DO SISTEMA mas somente se o "Indicador de Estado Z" está "Desligado".
(Este indicador pode ter sido posicionado, por exemplo, por uma instrução de comparação anterior.)

REGISTRADORES DE ESTADO CY,Z

Não são afetados.

RST

RST = RESTART = "REINICIE"

Estas instruções ASSEMBLER são transformadas pelo Compilador em Instruções do Z-80, cuja função é semelhante à função das instruções CALL, efetuando também uma "Chamada" a um certo endereço de memória.
A diferença está em que elas ocupam apenas um BYTE em linguagem de máquina e cada uma corresponde diretamente a um endereço pré-determinado (não há operando de endereço).

```
0010 I01:     RST  #10     (#D7)
0020 I02:     RST  #18     (#DF)
```

0010 Efetua "Chamada" (CALL) ao endereço de memória #0010, "Salvando" na PILHA DO SISTEMA o endereço de Retorno (endereço da Instrução I02).
0020 Efetua "Chamada" (CALL) ao endereço de memória #0018, "Salvando" na PILHA DO SISTEMA o endereço de Retorno (instrução seguinte à instrução RST).

INDICADORES DE ESTADO CY,Z

Não são afetados.

SBC

SBC = SUBTRACT WITH CARRY = "SUBTRAIA COM TRANSPORTE"

Estas Instruções ASSEMBLER são transformadas em Instruções Z-80 que efetuam a operação de "Subtrair" de uma Área de Trabalho primeiramente um Valor fornecido e, em seguida, o valor do Indicador de Carry-Status (0 ou 1).

Estas instruções podem trabalhar com Pares de Registradores, comportando neste caso valores entre #0000 e #FFFF.

Os Indicadores de Estado são posicionados.

```
0010            SBC  A,(HL)     (#9E)
0020            SBC  A,H        (#9C)
0030            SBC  HL,DE      (#ED #52)
```

0010 Subtrai de REG-A o valor atual do Indicador CY (que pode ser 1 ou 0, Ligado ou Desligado), e também o valor do BYTE contido no endereço apontado por REG-HL.
0020 Subtrai de REG-A o valor atual de CY,e também o conteúdo de REG-H.
0030 Subtrai de REG-HL o valor atual de CY (1 ou 0) e também o valor de REG-DE.

INDICADORES DE ESTADO CY,Z

Resultado menor do que Zero : C,NZ
Resultado igual a Zero : NC,Z
Resultado maior do que Zero : NC,NZ

SCF

SCF= SET CARRY-FLAG =
"LIGUE INDICADOR-DE-TRANSPORTE"

Esta Instrução ASSEMBLER é transformada pelo Programa Compilador em uma Instrução Z-80 cuja função é "Ligar o Indicador de Estado CY" (CARRY-FLAG - INDICADOR-DE-TRANSPORTE).

```
0010 B20:       SCF          (#37)
```

0010 "Liga" o Idicador CY (posiciona o BIT-0 de REG-F com o valor 1).

INDICADORES DE ESTADO CY,Z

CY = C (Ligado)
(O Indicador Z não é afetado.)

SUB

SUB = SUBTRACT = "SUBTRAIA"

Estas instruções ASSEMBLER são transformadas em Instruções de Máquina que solicitam ao Z-80 a "Subtração de um valor do REG-A"

Os Indicadores de Estado são posicionados para refletir as condições do resultado.

```
0010            SUB  #DA       (#D6 #DA)
0020            SUB  E         (#93)
0030            SUB  (HL)      (#96)
```

0010 Subtrai de REG-A o valor #DA.
0020 Subtrai de REG-A o valor contido em REG-E.
0030 Subtrai de REG-A o valor contido no BYTE apontado por REG-HL.

INDICADORES DE ESTADO CY,Z

Resultado menor do que Zero : C,NZ
Resultado igual a Zero : NC,Z
Resultado maior do que Zero : NC,NZ

DEMAIS INSTRUÇÕES

As demais Instruções ASSEMBLER/Z-80 são parecidas com as já descritas, ou fogem aos nossos objetivos.
Descreveremos as suas funções de maneira suscinta.

ADC

Soma em REG-A ou em REG-HL o valor atual do Indicador CY, mais um segundo Valor fornecido.
O funcionamento é semelhante ao da instrução SBC.
Exemplos:

```
0010        ADC  A,D   (#8A)
0020        ADC  HL,DE (#ED #5A)
```

BIT

Esta instrução "Testa" o valor de um BIT escolhido, de um registrador determinado ou da memória, verificando se o seu conteúdo é "0" ou "1".
Os Indicadores de Estado são posiconados para refletir os resultados do teste.
Exemplos :

```
0010        BIT  3,D     (#CB #5A)
0020        BIT  0,(HL)  (#CB #46)
```

CPD

Esta Instrução é semelhante à Instrução CPDR, exceto que REG-BC não é "Testado".

CPDR

Esta Instrução funciona de forma semelhante à instrução CPIR, exceto que REG-HL e REG-DE são "Subtraidos de 1" ao invéz de serem "Acrescidos de 1".

CPI

Esta Instrução funciona de forma semelhante à Instrução CPIR, exceto que REG-BC não é "Testado".

CPL

Esta Instrução "Complementa" o Valor do Acumulador (REG-A), ou seja, os BITS que contiverem "0" passarão a conter "1" e vice-versa.
Exemplo :

```
0010        CPL          (#2F)
```

DAA

Esta Instrução "Converte" o conteúdo de REG-A para o formato "Decimal Codificado em Binário".

DI

Esta Instrução "Bloqueia Interrupções", ou seja, não permite que a tarefa do Z-80 seja interrompida por outros dispositivos.

EI
Esta Instrução "Permite Interrupções", ou seja, passa a ser possível interromper a sequência normal de atividades do Z-80 para atender a dispositivos externos.

EXX
"Troca" o conteúdo dos Registradores HL, DE e BC com outras Áreas de Trabalho denominadas H'L', D'E' e B'C' que servem exclusivamente para "Salvar Temporariamente" os conteúdos destes Registradores.

HALT
Esta Instrução "Interrompe" a execução do programa corrente.

IM
Estas Instruções determinam o "Modo de Interrupção" que passa a estar ativo a partir deste momento.

IND
Esta Instrução executa um Input (Entrada) a partir de uma Porta de I/O para a memória, Decrementa Contador e Decrementa Endereço.

INDR
Semelhante a IND, repetida até que REG-B = 0.

INI
Semelhante a IND, exceto que Incrementa Endereço.

INIR
Semelhante a INI, repetida até que REG-B = 0.

LDD
Esta Instruão funciona de forma semelhante à Instrução LDDR, exceto que o conteúdo de REG-BC não é "Testado".

LDDR
Esta Instrução funciona de forma semelhante à Instrução LDIR, com a diferença que REG-HL e REG-DE são "Subtraídos de 1" ao invéz de serem "Acrescidos de 1".

LDI
Esta Instrução funciona de forma semelhante à Instrução LDIR, exceto que o conteúdo de REG-BC não é "Testado".

NEG
Esta Istrução Complementa o valor de REG-A.
Semelhante à Instrução CPL, exceto que posiciona os Indicadores de Estado.

NOP
Este código é interpretado pelo Z-80 como "Ausência de Instrução" (NOP = NO OPERATION - Sem Operação), e passa para o BYTE seguinte do programa.

OUTD
Esta instrução executa um Output (Saída) para uma porta de I/O, a partir de um endereço especificado, decrementa Contador e decrementa Endereço.

OTDR
Semelhante a OUTD, repetida até que REG-B = 0.

RES
Esta instrução "Desliga" o valor de um BIT escolhido, pertencente a um registrador determinado ou à memória, tornando-o "0".

Exemplos :

```
0010          RES  3,D      (#CB #9A)
0020          RES  0,(HL)   (#CB #8E)
```

RL/RR

(RL/RLA/RLC/RLCA/RLD)
(RR/RRA/RRC/RRCA/RRD)
Todas estas Instruções servem para "Girar" (ROTATE) o conteúdo de um Registrador ou de um BYTE da memória, isto é, "Deslocar" todos os seus BITS para a Esquerda (LEFT) ou para a Direita (RIGHT), de forma "Circular", com o "Primeiro BIT" sendo substituído pelo "Último BIT" (que "sai fora") ou pelo BIT CY.

SET
Esta Instrução faz a operação "Inversa" à da Instrução RES, ou seja, "Liga" o valor de um BIT escolhido, pertencente a um registrador determinado ou à memória, tornando-o "1".

SL/SR

(SLA/SRA/SRL)
Estas Instruções servem para "Deslocar (SHIFT) para a Direita (RIGHT) ou para a Esquerda (LEFT), os BITS de um Registrador ou de um BYTE da memória, inserindo ou não o BIT CY na operação.

INSTRUÇÕES ESPECIAIS

Existem algumas "Instruções" ASSEMBLER que servem somente para "Passar Informações ao Próprio Compilador", não sendo convertidas em Instruções de Máquina para o Z-80.
(Estas instruções podem variar de compilador para compilador.)

ORG #7000
Indica ao Compilador para que as Instruções seguintes convertidas para Linguagem Z-80 sejam colocadas a partir do endereço de memória #7000.

DEFB #3E,#FC,#00
Indica ao Compilador para que os próximos três BYTES do programa compilado contenham os valores #3E,#FC e #00.

DEFS 20
Indica ao Compilador para "Zerar" os próximos 20 BYTES do programa.
(Esta instrução é utilizada pelo BIT-BASIC para deixar BYTES Livres no programa.)

DEFM "Programa X"
Indica ao Compilador para colocar a constante "Programa X" nos próximos BYTES compilados.

;
Indica ao Compilador para tratar os caracteres seguintes como "Comentário", sem gerar qualquer "Código Objeto".

*E
Indica ao Compilador para "Saltar de Página" quando estiver listando o programa compilado.

CAPÍTULO IV

A P Ê N D I C E S

01 - VARIÁVEIS BIOS/BASIC

A seguir descreveremos as principais "Áreas de Trabalho" do BIOS e do BASIC.
Você poderá utilizar o seu conhecimento de ASSEMBLER Z-80 para analisar ou mesmo modificar o conteúdo destes campos, ou para implementar novas funções no BIT-BASIC.
É mostrado o endereço da Área de Trabalho na memória RAM, o seu nóme e o seu tamanho (número de BYTES).

#F39A USRTAB (20) - BYTES utilizados para salvar os endereços correspondentes aos comandos DEFUSR0=&HXXXX a DEFUSR9=&HYYYY (dois BYTES para cada um)
#F3AE LINL40 (1) - Número de BYTES da linha BASIC, no modo SCREEN0
#F3AF LINL32 (1) - Idem, modo SCREEN1
#F3B0 LINLEN (1) - Idem, no modo "Atual"
#F3B1 CRTCNT (1) - Número de linhas da Tela utilizadas p/BASIC
#F3B3 A #F3DA (80)- Endereços de início das "Tabelas" para o VDP (dois BYTES para cada endereço)
#F3DB CLIKSW (1) - Indicador de "Clic" do Teclado (#00 ou #FF)
#F3DC CSRY (1) - Posição "Y" do CURSOR (Vertical-Linha)
#F3DD CSRX (1) - Posição "X" do CURSOR (Horizontal-Coluna)
#F3DE CNSDFG (1) - Chave de "Mostra/Não Mostra" constantes correspondentes às Teclas de Função.
#F3E8 TRGFLG (1) - Indicador "Botões de Tiro" do Joy-Stick
#F3E9 FORCLR (1) - Cor dos caracteres na Tela
#F3EA BAKCLR (1) - Cor de "Fundo" da Tela
#F3EB BDRCLR (1) - Cor da "Borda" da Tela
#F3F8 PUTPNT (2) - Endereço "ATÉ" dos caracteres no KEY-BUFFER
#F3FA GETPNT (2) - Endereço "DE" dos caracteres no KEY-BUFFER
#F414 ERRFLG (1) - "Número do Erro" encontrado pelo BIOS/BASIC (variável ERR do BASIC)
#F41F KBUF (318)- "CRUNCH BUFFER" "ÁREA DE TRABALHO PARA TRITURAÇÃO" - Área para Montar/Desmontar os comandos processados
#F55E BUF (258)- "BUFFER BASIC" - Área de Trabalho onde a Linha digitada pelo usuário é colocada pela rotina PINLIN, antes de ser processada pelo BASIC
#F672 MEMSIZ (2) - Posição "Mais Alta" da Memória
#F674 STKTOP (2) - Endereço de início da PILHA DO SISTEMA
#F676 TXTTAB (2) - Endereço de início do "TEXTO BASIC" (início do Programa BASIC)
#F6AA AUTFLG (1) - Indica se o modo AUTO do BASIC está ou não "Ativo" (#00 ou #FF)
#F6AB AUTLIN (2) - Número da Linha Inserida pelo AUTO
#F6AD AUTINC (2) - Incremento do comando AUTO
#F6B3 ERRLIN (2) - Número da Linha onde ocorreu o Erro (Variável ERL do BASIC)
#F6B9 ONELIN (2) - Número da Linha para executar em caso de Erro (atualizado por ONERRORGOTO do BASIC)

```
#F6C2 VARTAB (2)   - Endereço de início da Area de Variáveis
                     Simples do BASIC
#F6C4 ARYTAB (2)   - Endereço de início da Tàbela de Array
#F6C6 STREND (2)   - Fim da memória em uso pelo BASIC
#F6C8 DATPTR (2)   - "Pointer" (Apontador) para o DADO (DATA)
                     seguinte a ser lido por um comando READ
                     (atualizado por um comando RESTORE)
#F7C4 TRCFLG (1)   - Atualizada pelo comando TRON (Trace)
#F87F FNKSTR (160)- Dez áreas de 16 BYTES cada uma onde
                     são guardadas as constantes das Teclas de
                     Função (PFKEYS)
#FBF0 KEYBUF (40) - Área de Trabalho (BUFFER) do Teclado
#FC48 BOTTOM (2)   - Início da RAM utilizada pelo BASIC (#8001)
#FC4A HIMEM  (2)   - Final da RAM disponível para o BASIC
#FCA8 INSFLG (1)   - Indicador de modo INS "Ligado/Desligado"
#FCA9 CSRSW  (1)   - Chave para mostrar ou não o CURSOR
#FCAA CSTYLE (1)   - Estilo do CURSOR
```

02 - PONTOS DE ENTRADA BIOS/BASIC

A seguir mostraremos os endereços de início (Pontos de Entrada) para diversas rotinas do BIOS/BASIC, a função de cada uma, as condições de entrada (ENT), as condições de saída (SAI) e as áreas de trabalho por elas utilizadas/modificadas (MOD).
Algumas destas rotinas são utilizadas pelo BIT-BASIC (e detalhadas durante a descrição deste programa).
Você poderá utilizar estas rotinas em seus programas ASSEMBLER ou para incluir novas funções no BIT-BASIC.

——— #0000 CHKRAM ———

Verifica RAM e posiciona SLOTS (Inicialização)

——— #0004 CGTABL ———

Endereço da Tabela do Gerador de Caracteres

——— #0006 VDP.DR ———

Endereço Registrador VDP (Leitura)

——— #0007 VDP.DW ———

Endereço Registrador VDP (Gravação)

——— #0008 SYNCHR ———

Verifica se Caráter do comando é válido - Se sim, desvia para CHRGTR, se não, Erro de Sintaxe
ENT - HL=Endereço do caráter
SAI - A =Código do caráter
HL=Endereço próximo caráter
CY=1 se caráter numérico
Z =1 se fim do comando

——— #000C RDSLT ———

Seleciona SLOT (Banco Memória ROM/RAM), lê BYTE
ENT - A =Seleciona SLOT (FxxxSSPP)
F = 1 significa que SS foi especificado
SS = Número SLOT secundário
PP = Número SLOT primário
HL=Endereço de memória a ser lido
SAI - A =Conteúdo do endereço de memória lido
MOD - AF,BC,DE (Interrupções Desabilitadas)

——— #0010 CHRGTR ———

Lê próximo caráter do TEXTO BASIC
ENT - HL=Endereço caráter a ser lido
SAI - A =Caráter lido
HL=Endereço próximo caráter
CY=1 se o caráter é numérico
Z =1 se fim do Texto (#00)

——— #0014 WRSLT ———

Seleciona SLOT (Banco de Memória) e grava BYTE
ENT - A =Seleciona SLOT (FxxxSSPP)
F = 1 significa que SS foi especificado
SS = Número SLOT secundário
PP = Número SLOT primário
HL=Endereço de memória a ser gravado
E =BYTE a ser gravado
MOD - AF,BC,DE (Interrupções Desabilitadas)

——— #0018 OUTDO ———

Saída de um BYTE para o "Dispositivo Corrente"
ENT - A =BYTE a ser remetido p/saída
PTRFIL (#F864)
PTRFLG (#F416)

——— #001C CALSLT ———

Chamada (CALL) entre SLOTS
ENT - IY=Seleciona SLOT (FxxxSSPP)
F=1 significa que SS foi especificado
SS = Número SLOT secundário
PP = Número SLOT primário
IX=Endereço a ser chamado
SAI - Indeterminado
MOD - Indeterminado (Interrupções Desabilitadas)

——— #0020 DCOMPR ———

Compara REG-HL com REG-DE
ENT - HL=Qualquer conteúdo
DE=Qualquer conteúdo
SAI - C,NZ - DE>HL
NC,Z - DE=HL
NC,NZ - DE<HL
MOD - AF

——— #0024 ENASLT ———

Habilita (Ativa) SLOT
ENT - A =Seleciona SLOT (FxxxSSPP)
F=1 significa que SS foi especificado
SS = Número SLOT secundário
PP = Número SLOT primário
HL=Endereço de memória
MOD - Todos Registradores (Interrupções Desabilitadas)

——— #0028 GETYPR ———

Fornece o tipo de FAC
ENT - FAC
SAI - Indicadores "Registrador F"
MOD - AF

——— #002B VERMSX ———

Cinco BYTES onde está armazenado o "Número de Versão" do MSX (primeira versão contém Zeros).

——— #0030 CALLF ———

Efetua chamada (CALL) entre SLOT'S
ENT - Instrução ASSEMBLER = #F7 #SS #EE #EE
#F7 = RST #30 (CALL #0030)
#SS = Número do SLOT
#EEEE = Endereço a ser chamado
SAI - Indeterminado
MOD - Indeterminado

——— #0038 KEYINT ———

Executa procedimentos de Interrupção de Hardware

AS ROTINAS SEGUINTES SÃO UTILIZADAS PARA INICIALIZAÇÃO DE I/O

——— #003B INITIO ———

Executa inicialização dos dispositivos
MOD - Todos

——— #003E INIFNK ———

Inicializa conteúdo Teclas de Função (PFKEYS)
MOD - Todos

AS ROTINAS SEGUINTES SÃO UTILIZADAS PARA ACESSAR O VDP (VÍDEO DISPLAY PROCESSOR - PROCESSADOR DE TELA)

#0041 DISSCR

Desabilita Tela (não mostra imagem)
MOD - AF,BC

#0044 ENASCR

Habilita Tela (mostra imagem)
MOD - AF,BC

#0047 WRTVDP

Grava um BYTE num Registrador do VDP
ENT - C =Número do Registrador
B =Dado a ser gravado
MOD - AF,BC

#004A RDVRM

Lê um BYTE da VRAM
ENT - HL=Endereço da VRAM a ser obtido
SAI - A =BYTE da VRAM
MOD - AF

#004D WRTVRM

Grava um BYTE na VRAM
ENT - HL=Endereço da VRAM
A =BYTE a ser gravado
MOD - AF

#0050 SETRD

Posiciona VDP para Leitura
ENT - HL
MOD - AF

#0053 SETWRT

Posiciona VDP para Gravação
ENT - HL
MOD - AF

#0056 FILVRM

Preenche um trecho da VRAM com um BYTE determinado
ENT - HL=Endereço inicial
BC=Tamanho da área a ser preenchida
A =BYTE a ser movido
MOD - Todos

#0059 LDIRMV

Copia um bloco da memória VRAM para memória RAM
ENT - HL=Endereço inicial VRAM
DE=Endereço inicial RAM
BC=Tamanho do Bloco a ser copiado
MOD - Todos

#005C LDIRVM

Copia um bloco da RAM para a VRAM
ENT - HL=Endereço inicial RAM
DE=Endereço inicial VRAM
BC=Tamanho do Bloco a ser copiado
MOD - Todos

#005F CHGMOD

Posiciona o modo de operação do VDP
ENT - Conteúdo Área de Trabalho SCRMOD (#FCAF)
MOD - Todos

#0062 CHGCLR

Muda cores da Tela
ENT - FORCLR=Cor dos caracteres na Tela (#F3E9)
BAKCLR=Cor de Fundo da Tela (#F3EA)
BDRCLR=Cor da Borda da Tela (#F3EB)
MOD - Todos

#0066 NMI

Procedimentos de "Interrupção não Mascarável"

#0069 CLRSPR

Inicializa todos os SPRITES (Padrão Zeros, nomes, cor das letras, posição vertical 209)
ENT - SCRMOD=SCREEN-MODE (Modo da Tela - #FCAF)
MOD - Todos

PARA AS ENTRADAS SEGUINTES VALEM AS CONVENÇÕES
TXT = Modo TEXTO (SCREEN0 - 40X24)
T32 = Modo TEXTO (SCREEN1 - 32X24)
GRP = Modo ALTA RESOLUÇÃO (SCREEN2)
MLT = Modo MULTICOR (SCREEN3)
NAM = Tabela de Nomes (Caracteres)
CGP = Tabela de Padrões
COL = Tabela de Cores
ATR = Tabela de Atributos de SPRITES
PAT = Tabela de Padrões de SPRITES

#006C INITXT

Inicializa Tela para modo TEXTO (SCREEN0)
ENT - TXTNAM (#F3B3),TXTCGP (#F3B7)
MOD - Todos

#006F INIT32

Inicializa Tela para modo TEXTO (SCREEN1)
ENT - T32NAM (#F3BD), T32CGP (#F3C1), T32COL (#F3BF), T32ATR (#F3C3), T32PAT (#F3C5)
MOD - Todos

#0072 INIGRP

Inicializa Tela para modo ALTA RESOLUÇÃO (SCREEN2)
ENT - GRPNAM (#F3C7), GRPCGP (#F3CB), GRPCOL (#F3C9), GRPATR (#F3CD), GRTPAT (#F3CF)
MOD - Todos

#0075 INIMLT

Inicializa Tela para modo MULTICOR (SCREEN3)
ENT - MLTNAM (#F3D1), MLTCGP (#F3D5), MLTCOL (#F3D3), MLTATR (#F3D7), MLTPAT (#F3D9)
MOD - Todos

#0078 SETTXT

Posiciona VDP para modo TEXTO (SCREEN0)
ENT - TXTNAM, TXTCGP
MOD - Todos

#007B SETT32

Posiciona VDP para modo TEXTO (SCREEN1)
ENT - T32NAM, T32CGP, T32COL, T32ATR, T32PAT
MOD - Todos

#007E SETGRP

Posiciona VDP para modo ALTA RESOLUÇÃO (SCREEN2)
ENT - GRPNAM, GRPCGP, GRPCOL, GRPATR, GRPPAT
MOD - Todos

——— #0081 SETMLT ———

Posiciona VDP para modo MULTICOR (SCREEN3)
ENT - MLTNAM, MLTCGP, MLTCOL, MLTATR, MLTPAT
MOD - Todos

——— #0084 CALPAT ———

Fornece endereço da Tabela de Padrões de SPRITE
ENT - A =Identificação (Número) do SPRITE
SAI - HL=Endereço da Tabela de Padrões
MOD - AF,DE,HL

——— #0087 CALATR ———

Fornece endereço Tabela de Atributos de SPRITE
ENT - A =Identificação (Número) do SPRITE
SAI - HL=Endereço da Tabela de Atributos
MOD - AF,DE,HL

——— #008A GSPSIZ ———

Fornece Tamanho atual dos SPRITES
ENT - Nada
SAI - A =Tamanho dos SPRITES (Número de BYTES)
CY=1 Se SPRITES 16*16
MOD - AF

——— #008D GRPPRT ———

Coloca caráter na Tela Gráfica
ENT - A =Caráter a ser apresentado

AS ENTRADAS SEGUINTES SÃO UTILIZADAS PARA ACESSAR O PSG (PROGRAMMABLE SOUND GENERATOR - GERADOR DE SONS PROGRAMÁVEL)

——— #0090 GICINI ———

Inicializa PSG para PLAY
MOD - Todos

——— #0093 WRTPSG ———

Envia um dado para um Registrador do PSG
ENT - A =Número do Registrador
E =Dado a ser enviado

——— #0096 RDPSG ———

Lê dado de um Registrador do PSG
ENT - A =Número do Registrador
SAI - A =Dado lido do PSG

——— #0099 STRTMS ———

Executa atividade "de fundo" para o PLAY
MOD - Todos

AS ENTRADAS SEGUINTES SÃO UTILIZADAS PARA ACESSAR O TECLADO, TELA E IMPRESSORA

——— #009C CHSNS ———

Verifica o BUFFER do Teclado
SAI - Z = 1 (Z) Se há algum caráter no BUFFER
MOD - AF

——— #009F CHGET ———

Aguarda até que um caráter seja digitado fornecendo o seu código.
SAI - A =Código do caráter digitado no teclado
MOD - AF

——— #00A2 CHPUT ———

Envia um caráter para a Tela
ENT - A =Caráter a ser apresentado na Tela

——— #00A5 LPTOUT ———

Envia um caráter para a Impressora
ENT - A =Caráter a ser enviado
CY=1 (C) se não houve sucesso
MOD - F

——— #00A8 LPTSTT ———

Verifica estado da Impressora
SAI - A=#FF e Z=0 (NZ) se a Impressora "Pronta"
A=#00 e Z=1 (Z) se Impressora NÃO "Pronta"
MOD - AF

——— #00AB CNVCHR ———

Verifica BYTE controle gráfico e converte código
ENT - A =Caráter
SAI - NC=BYTE de controle gráfico
C,Z=Código gráfico convertido
C,NZ=Código não convertido
MOD - AF

——— #00AE PINLIN ———

Obtém linha digitada via Teclado, até o acionamento da Tecla RETURN ou CONTROL+STOP.
Coloca linha digitada no BUFFER (#F55E).
SAI - HL=Endereço do BUFFER menos 1 (#F55D)
CY=1 (C) CONTROL-STOP pressionadas
MOD - Todos

——— #00B1 INLIN ———

Idem PINLIN, exceto quando modo AUTO ativo

——— #00B4 QINLIN ———

Coloca "?" na Tela e desvia para INLIN

——— #00B7 BREAKX ———

Verifica se as teclas CONTROL+STOP estão pressionadas
SAI - CY=1 (C) se CONTROL+STOP estão pressionadas
MOD - AF

——— #00BA ISCNTC ———

Verifica teclas SHIFT-STOP
#00BD CKCNTC - Idem ISCNTC, usada pelo BASIC

——— #00C0 BEEP ———

Emite um som de "Alarme" (BIP)
MOD - Todos

——— #00C3 CLS ———

"Limpa" a Tela
MOD - AF,BC,DE

——— #00C6 POSIT ———

Coloca o CURSOR em uma posição escolhida da Tela
ENT - H =Coluna (Horizontal)
L =Linha (Vertical)
MOD - AF

——— #00C9 FNKSB ———

Mostra o conteúdo da Teclas de Função na Tela (PFKEYS), se indicador "Ligado" (FNKFLG-#FBCE)

——— #00CC ERAFNK ———

"Apaga" Teclas de Função (PFKEYS)
MOD - Todos

——— #00CF DSPFNK ———

Mostra Teclas de Função na Tela (PFKEYS)
MOD - Todos

--------------- #00D2 TOTEXT ---------------
Força a Tela para o modo TEXTO
MOD - Todos

AS ENTRADAS SEGUINTES SÃO UTILIZADAS PARA ACESSAR OS PERIFÉRICOS DE JOGOS

--------------- #00D5 GTSTCK ---------------
Devolve o Estado corrente do JOY-STICK
ENT - A =Identificação do JOY-STICK
SAI - A =Direção atual do JOY-STICK
MOD - Todos

--------------- #00D8 GTTRIG ---------------
Devolve o estado corrente do "Botão de Tiro"
ENT - A =Identificação do Botão de Tiro
SAI - A =#00 indica botão "Não pressionado"
A =#FF indica botão "Pressionado"
MOD - AF

--------------- #00DB GTPAD ---------------
Devolve o Estado corrente do PADDLE
ENT - A =Identificação do PADDLE
SAI - A =Estado atual do PADDLE
MOD - Todos

--------------- #00DE GTPDL ---------------
Devolve o valor corrente do PADDLE
ENT - A =Identificação PADDLE
SAI - A =Valor atual do PADDLE
MOD - Todos

AS ENTRADAS SEGUINTES SÃO UTILIZADAS PARA ACESSAR O GRAVADOR

--------------- #00E1 TAPION ---------------
Liga motor do gravador e lê "Cabeçalho" da fita
SAI - CY=1 (C) se não houve sucesso
MOD - Todos

--------------- #00E4 TAPIN ---------------
Lê um BYTE da fita
SAI - A =BYTE lido
CY=1 (C) leitura interrompida
MOD - Todos

--------------- #00E7 TAPIOF ---------------
Encerra leitura da fita

--------------- #00EA TAPOON ---------------
Liga motor do gravador e grava "Cabeçalho"
ENT - A =0 Se cabeçalho "Curto" desejado
A<>0 Se cabeçalho "Longo" desejado
SAI - CY=1 (C) se gravação interrompida
MOD - Todos

--------------- #00ED TAPOUT ---------------
Grava um BYTE na fita
ENT - A =BYTE a ser gravado
SAI - CY=1 (C) se gravação interrompida
MOD - Todos

--------------- #00F0 TAPOOF ---------------
Encerra a gravação da fita

--------------- #00F3 STMOTR ---------------
Liga/Desliga o motor do gravador
ENT - A =#00 Para desligar o motor
A =#01 Para ligar o motor
A =#255 Para inverter o estado do motor

AS ENTRADAS SEGUINTES SÃO PARA TRATAMENTO DE FILAS

--------------- #00F6 LFTQ ---------------
Retorna quantos BYTES existem na fila

--------------- #00F9 PUTQ ---------------
Coloca um BYTE na fila

AS ENTRADAS SEGUINTES SÃO UTILIZADAS PELAS ROTINAS DE TRATAMENTO DE TELA GRÁFICA (GENGRP E ADVGRP)

--------------- #00FC RIGHTC ---------------
Move um PIXEL para a direita

--------------- #00FF LEFTC ---------------
Move um PIXEL para a esquerda

--------------- #0102 UPC ---------------
Move um PIXEL para cima

--------------- #0105 TUPC ---------------
Move um PIXEL para cima

--------------- #0108 DOWNC ---------------
Move um PIXEL para baixo

--------------- #010B TDOWNC ---------------
Move um PIXEL para baixo

--------------- #010E SCALXY ---------------
Coordenadas das Escalas X e Y

--------------- #0111 MAPXYC ---------------
Mapeia coordenadas X/Y para o endereço físico

--------------- #0114 FETCHC ---------------
Carrega endereço físico atual e padrão máscara
SAI - HL=Endereço físico
A =Padrão de Máscara

--------------- #0117 STOREC ---------------
Armazena endereço físico e padrão de máscara
ENT - HL=Endereço físico
A =Padrão de Máscara

--------------- #011A SETATR ---------------
Posiciona o BYTE de atributo

--------------- #011D READC ---------------
Lê atributos do PIXEL corrente

--------------- #0120 SETC ---------------
Seta PIXEL corrente com atributo fornecido

--------------- #0123 NSETCX ---------------
Posiciona PIXELS horizontalmente

--------------- #0126 GTASPC ---------------
Retorna "Razão de Aspecto"
SAI - DE,HL

--------------- #0129 PNTINI ---------------
Inicialização para PAINT

--------------- #012C SCANR ---------------
Pesquisa PIXELS para a direita

--------------- #012F SCANL ---------------
Pesquisa PIXELS para a esquerda

AS ENTRADAS SEGUINTES EXECUTAM FUNÇÕES DE USO GERAL

--------------- #0132 CHGCAP ---------------
Muda estado da lâmpada CAPS-LOCK (Liga/Desliga)
ENT - A =#00 Para apagar a lâmpada
A<>#00 Para acender a lâmpada
MOD - AF

```
--------------- #0135 CHGSND -------------------
Muda o estado de BIT de som
ENT    - A =#00 Para desligar o BIT de som
         A<>#00 Para ligar o BIT de som
MOD    - AF
--------------- #0138 RSLREG -------------------
Retorna BYTE corrente enviado ao SLOT primário
SAI    - A =BYTE lido
--------------- #013B WSLREG -------------------
Grava BYTE no registrador do SLOT primário
ENT    - A =BYTE a ser gravado
--------------- #013E RDVDP --------------------
Lê registrador de Estado do VDP
SAI    - A=Conteúdo do Registrador
MOD    - AF
--------------- #0141 SNSMAT -------------------
Retorna estado uma "Fila" da Matriz do Teclado
ENT    - A =Número da Linha
SAI    - A =BITS "0" correspondem Teclas acionadas
MOD    - AF
--------------- #0144 PHYDIO -------------------
Operações para dispositivos de armazenamento
--------------- #0147 FORMAT -------------------
Inicialização dispositivos de armazenamento
--------------- #014A ISFLIO -------------------
Verifica se está sendo executada operação de I/O
MOD    - AF
--------------- #014D OUTDLP -------------------
Envia um caráter para a Impressora
ENT    - A =BYTE a ser enviado
MOD    - F
NOTAS  - Esta rotina difere de  LPTOUT  (#00A5)  no
seguinte
   1-TAB's são convertidos em espaços
   2-HIRAGANA e Símbolos Gráficos são convertidos
     quando a impressora não é MSX
   3-Quando   a   impressão   é   interrompida,   é
     apresentada a mensagem de "ERRO DE I/O"
--------------- #0150 GETVCP -------------------
Rotina para execução de música "de Fundo"
--------------- #0153 GETVC2 -------------------
Rotina para execução de música "de Fundo"
--------------- #0156 KILBUF -------------------
"Limpa" BUFFER do Teclado
MOD    - HL
--------------- #0159 CALBAS -------------------
Executa Inter-SLOT CALL's internamente ao BASIC
ENT    - IX=Endereço
SAI    - Indeterminado
MOD    - Indeterminado
------------------------------------------------
A SEGUIR EXISTEM 90 BYTES RESERVADOS  PARA  O  BIOS
(VOCÊ NÃO PODE UTILIZÁ-LOS POIS PERTENCEM À ROM)
------------------------------------------------
```

03 — GANCHOS BIOS/BASIC

No ítem I-02 você tem uma descrição do funcionamento e utilidade dos GANCHOS do MSX.
Neste Apêndice relacionamos os endereços dos GANCHOS chamados pelas diversas rotinas do BIOS/BASIC.
O programa BIT-BASIC utiliza alguns destes GANCHOS para "Interceptar" e introduzir modificações funcionais nas rotinas corresondentes.
Você também pode montar programas ASSEMBLER (ou mesmo programas BASIC) para utilizar estes GANCHOS.

Cada GANCHO ocupa CINCO BYTES e serão mostrados o seu Endereço Inicial, o nome da rotina do BIOS/BASIC que efetua desvio para este endereço e o endereço da Instrução CALL que faz este desvio (entre parêntesis).
Várias destas rotinas estão descritas no apêndice 02 (Pontos de Entrada BIOS/BASIC).

```
#FD9A KEYI  (#00C4) Tratamento de Interrupções
#FD9F TIMI  (#0C53) Tratamento de Interrupções
#FDA4 CHPU  (#08C0) CHPUT (Envia um caráter para a Tela)
#FDA9 DSPC  (#09E6) DSPCSR (Mostra o CURSOR)
#FDAE ERAC  (#0A33) ERACSR (Apaga o CURSOR)
#FDB3 DSPF  (#0B2B) DSPFNK (Mostra PFKEYS)
#FDB8 ERAF  (#0B15) ERAFNK (Apaga PFKEYS)
#FDBD TOTE  (#0842) TOTEXT (Força Tela para modo TEXTO)
#FDC2 CHGE  (#10CE) CHGET  (Input caráter)
#FDC7 INIP  (#071E) INIPAT (Inicializa Padrão Caracteres VRAM)
#FDCC KEYC  (#1025) KEYCOD (Codificador do Teclado)
#FDD1 KEYA  (#0F10) KEYEASY (Tecla Fácil)
#FDD6 NMI   (#1398) NMI     (Interrupção Não Mascarável)
#FDDB PINL  (#23BF) PINLIN (Obtém linha digitada)
#FDE0 QINL  (#23CC) QINLIN ("Interrogação" mais PINLIN)
#FDE5 INLI  (#23D5) INLIN  (PINLIN mais AUTO)
#FDEA ONGO  (#7810) ONGOTP (ON GO TO)
#FDEF DSKO  (#7C16) DSKO$  (Gravação Bloco em Diskette)
#FDF4 SETS  (#7C1B) SETS   (Posiciona Atributos Diskette)
#FDF9 NAME  (#7C20) NAME   (Rotina RENAME para Diskette)
#FDFE KILL  (#7C25) KILL   (Rotina "Apaga Arquivo" Diskette)
#FE03 IPL   (#7C2A) IPL    (Carrega Progr.Inicial p/Diskette)
#FE08 COPY  (#7C2F) COPY   (Copia arquivos Diskette)
#FE0D CMD   (#7C34) CMD    (Rotina COMMAND p/Diskette)
#FE12 DSKF  (#7C39) DSKF   (Rotina Disco Livre p/Diskette)
#FE17 DSKI  (#7C3E) DSKI$  (Lê Bloco Diskette)
#FE1C ATTR  (#7C43) ATTR$  (Rotina de Atributos p/Diskette)
#FE21 LSET  (#7C48) LSET   (Posiciona Esquerda - Diskette)
#FE26 RSET  (#7C4D) RSET   (Posiciona Direita - Diskette)
#FE2B FIEL  (#7C52) FIELD  (Rotina CAMPO p/Diskette)
#FE30 MKI$  (#7C57) MKI$   (Conversão caracteres p/Diskette)
#FE35 MKS$  (#7C5C) MKS$   (Conversão caracteres p/Diskette)
#FE3A MKD$  (#7C61) MKD$   (Conversão caracteres p/Diskette)
#FE3F CVI   (#7C66) CVI    (Conversão caracteres p/Diskette)
#FE44 CVS   (#7C6B) CVS    (Conversão caracteres p/Diskette)
#FE49 CVD   (#7C70) CVD    (Conversão caracteres p/Diskette)
```

```
#FE4E GETP   (#6A93) GETPTR (Apontador Arquivo p/Diskette)
#FE53 SETF   (#6AB3) SETFIL (Posiciona Apontador Arquivo)
#FE58 NOFO   (#6AF6) NOFOR  (Cláusula NO FOR p/Diskette)
#FE5D NULO   (#6B0F) NULOPN (OPEN NULL p/Arquivo Diskette)
#FE62 NTFL   (#6B3B) NTFLO  (NOT FILE 0 p/Diskette)
#FE67 MERG   (#6B63) MERGE  (MERGE Arquivos Diskette)
#FE6C SAVE   (#6BA6) SAVE   (SAVE Arquivos Diskette)
#FE71 BINS   (#6BCE) BINSAV (BSAVE Arquivos Diskette)
#FE76 BINL   (#6BD4) BINLOD (BLOAD Arquivo Diskette)
#FE7B FILE   (#6C2F) FILES  (FILES - ARQUIVOS Diskette)
#FE80 DGET   (#6C3B) DGET   (GET p/Diskette)
#FE85 FILO   (#6C51) FILOU1 (FILE OUT 1 p/Diskette)
#FE8A INDS   (#6C79) INDSKC (Lê Caráter Diskette)
#FE8F RSLF   (#6CD8)        (Redireciona Drive Antigo Diskette)
#FE94 SAVD   (#6D03)        (Salva Drive Corrente Diskette)
#FE99 LOC    (#6D0F) LOC    (Localização Arquivo Diskette)
#FE9E LOF    (#6D20) LOF    (Tamanho Arquivo Diskette)
#FEA3 EOF    (#6D33) EOF    (Fim Arquivo Diskette)
#FEA8 FPOS   (#6D43) FPOS   (Posição Arquivo Diskette)
#FEAD BAKU   (#6E36) BAKUPT (BACK UP - CÓPIA Arquivo Diskette)
#FEB2 PARD   (#6F15) PARDEV (Verifica Nome Periférico)
#FEB7 NODE   (#6F33) NODEVN (Nome Padrão Periférico)
#FEBC POSD   (#6F37) POSDSK (Rotina POSSIBLY DISK)
#FEC1 DEVN   Sem Uso DEVNAM (Nome do Dispositivo)
#FEC6 GEND   (#6F8F) GENDSP (Acionador Geral de Dispositivos)
#FECB RUNC   (#629A) FUNC   (Rotina RUN CLEAR)
#FED0 CLEA   (#62A1) CLEARC (Rotina CLEAR)
#FED5 LOPD   (#62AF) LOPDFT (Posiciona Variáveis Padrão)
#FEDA STKE   (#62F0) STKERR (Erro de STACK - PILHA)
#FEDF ISFL   (#145F) ISFLIO (I/O Arquivo IS)
#FEE4 OUTD   (#1B46) OUTDO  (Rotina OUTDO - Saída caráter)
#FEE9 CRDO   (#7328) CRDO   (RETURN + LINE-FEED)
#FEEE DSKC   (#7374) DSKCHI (Entrada caráter Diskette)
#FEF3 DOGR   (#593C) DOGRPH (Rotina Gráfica)
#FEF8 PRGE   (#4039) PRGEND (Fim de Programa)
#FEFD ERRP   (#40DC) ERRPRT (Saída de Erros)
#FF02 ERRF   (#40FD)        (Idem)
#FF07 READ   (#4128) READY  (Rotina "Pronto" - OK)
#FF0C MAIN   (#4134) MAIN   (Entrada PRINCIPAL do BASIC)
#FF11 DIRD   (#41A8) DIRDO  (Executa COMANDO DIRETO)
#FF16 FINI   (#4237)        (Fim Entrada Principal)
#FF1B FINE   (#4247)        (Idem)
#FF20 CRUN   (#42B9)        (Codifica/Decodifica Linha)
#FF25 CRUS   (#4353)        (Idem)
#FF2A ISRE   (#437C)        (Idem)
#FF2F NTFN   (#43A4)        (Idem)
#FF34 NOTR   (#44EB)        (Idem)
#FF39 SNGF   (#45D1)        (Instrução FOR)
#FF3E NEWS   (#4601)        (Nova Instrução)
#FF43 GONE   (#4646)
#FF48 CHRG   (#4666)        (Rotina CHRGTR)
#FF4D RETU   (#4821)        (RETURN)
#FF52 PRTF   (#4A5E)        (PRINT)
#FF57 COMP   (#4A94)        (PRINT)
#FF5C FINP   (#4AFF)        (PRINT)
```

```
#FF61 TRMN   (#4B4D)         (Erro READ ou INPUT)
#FF66 FRME   (#4C6D)         (Analisador Expressões)
#FF6B NTPL   (#4CA6)         (Idem)
#FF70 EVAL   (#4DD9)         (Analisador Fatores)
#FF75 OKNO   (#4F2C)         (Idem)
#FF7A FING   (#4F3E)         (Idem)
#FF7F ISMI   (#51C3) ISMID$  (MID$)
#FF84 WIDT   (#51CC) WIDTH   (Largura linhas Tela)
#FF89 LIST   (#522E) LIST    (LIST programa BASIC)
#FF8E BUFL   (#532D) BUFLIN  (Rotina BUFFER LINE)
#FF93 FRQI   (#543F) FRQINT  (Conversão Inteiro)
#FF98 SCNE   (#5514)         (Apontador Número de Linha)
#FF9D FRET   (#67EE) FRETEMP (Rotina FREE UP TEMPORARIES)
#FFA2 PTRG   (#5EA9) PTRGET  (Rotina POINTER GET)
#FFA7 PHYD   (#148A) PHYDIO  (I/O Físico Diskettes)
#FFAC FORM   (#148E) FORMAT  (Formata Diskettes)
#FFB1 ERRO   (#406F) ERROR   (Tratamento de Erros)
#FFB6 LPTO   (#085D) LPTOUT  (Envia Caráter p/Impressora)
#FFBB LPTS   (#0884) LPTSTT  (Verifica Estado Impressora)
#FFC0 SCRE   (#79CC) SCREEN  (Entrada comando SCREEN)
#FFC5 PLAY   (#73E5) PLAY    (Entrada comando PLAY)

#FFCA até #FFFE      Área não utilizada pelo BIOS/BASIC
```

04 - PROGRAMA BIT-BASIC COMPILADO

```
FFD9              10            ORG  #FFD9
                  20 ;
FFD9  C1          30 DESVIO:    POP  BC
FFDA  CDC223      40            CALL #23C2
FFDD  D8          50            RET  C
FFDE  E5          60            PUSH HL
FFDF  D7          70            RST  #10
FFE0  21F5FF      80            LD   HL,DV09
FFE3  010400      90            LD   BC,#04
FFE6  A7          100           AND  A
FFE7  EDB1        110           CPIR
FFE9  E1          120           POP  HL
FFEA  C0          130           RET  NZ
FFEB  C1          140 DV01:     POP  BC
FFEC  010070      150           LD   BC,INICIO
FFEF  C5          160           PUSH BC
FFF0  3EFC        170 DV02:     LD   A,#FC
FFF2  D3A8        180 DV03:     OUT  (#A8),A
FFF4  C9          190           RET
FFF5  3C5B3D2E    200 DV09:     DEFB #3C,#5B,#3D,#2E
                  210 ;
7000              220           ORG  #7000
                  230 ;
7000  3A477B      240 INICIO:   LD   A,(T19)
7003  FEC3        250           CP   #C3
7005  CAC477      260           JP   Z,CML02
7008  E5          270           PUSH HL
7009  CD5B70      280 A01:      CALL RST10
700C  CD7370      290           CALL MINUSC
700F  77          300           LD   (HL),A
7010  FE00        310           CP   #00
7012  20F5        320           JR   NZ,A01
7014  E1          330           POP  HL
7015  CD5B70      340           CALL RST10
7018  FE3C        350           CP   #3C
701A  CA6B71      360           JP   Z,LISTAPG
701D  FE5B        370           CP   #5B
701F  CAD172      380           JP   Z,VOLTLIN
7022  FE3D        390           CP   #3D
7024  CAE575      400           JP   Z,VERCTE
7027  CD5B70      410           CALL RST10
702A  CAB970      420           JP   Z,RET03
702D  DAE478      430           JP   C,LISTLIN
7030  FE7A        440           CP   #7A
7032  CA0075      450           JP   Z,NOVOBAS
7035  FE63        460           CP   #63
7037  CA8476      470           JP   Z,COPMOV
703A  FE6D        480           CP   #6D
703C  CA8476      490           JP   Z,COPMOV
703F  FE3D        500           CP   #3D
7041  CADF70      510           JP   Z,ENCERRA
7044  C30478      520           JP   SINTAXE
7047              530           DEFS 20
                  540 *E
705B  23          550 RST10:    INC  HL
705C  7E          560           LD   A,(HL)
705D  FE00        570           CP   #00
705F  C8          580           RET  Z
7060  FE30        590           CP   #30
7062  3F          600           CCF
7063  D0          610           RET  NC
7064  FE3A        620           CP   #3A
7066  D8          630           RET  C
7067  B7          640           OR   A
7068  C9          650           RET
7069              660           DEFS 10
                  670 ;
7073  FE41        680 MINUSC:   CP   #41
7075  D8          690           RET  C
7076  FE5B        700           CP   #5B
7078  D0          710           RET  NC
7079  C620        720           ADD  A,#20
707B  C9          730           RET
707C              740           DEFS 10
                  750 ;
7086  11F0FF      760 CALL:     LD   DE,DV02
7089  D5          770           PUSH DE
708A  C5          780 JUMP:     PUSH BC
708B  3EF0        790           LD   A,#F0
708D  C3F2FF      800           JP   DV03
7090              810           DEFS 10
                  820 ;
709A  210D00      830 RET01:    LD   HL,#000D
709D  22F0FB      840 RET02:    LD   (#FBF0),HL
70A0  21F0FB      850           LD   HL,#FBF0
70A3  22FAF3      860           LD   (#F3FA),HL
70A6  23          870           INC  HL
70A7  22F8F3      880           LD   (#F3F8),HL
70AA  013741      890           LD   BC,#4137
70AD  18DB        900           JR   JUMP
70AF              910           DEFS 10
                  920 ;
70B9  215DF5      930 RET03:    LD   HL,#F55D
70BC  017341      940           LD   BC,#4173
70BF  C38A70      950           JP   JUMP
70C2              960           DEFS 10
                  970 ;
70CC  CD5601      980 RET04:    CALL #0156
70CF  013741      990           LD   BC,#4137
70D2  C38A70      1000          JP   JUMP
70D5              1010          DEFS 10
                  1020 ;
70DF  3EC9        1030 ENCERRA: LD   A,#C9
70E1  32DBFD      1040          LD   (#FDDB),A
70E4  18E6        1050          JR   RET04
70E6              1060          DEFS 10
```

```
                1070 *E
70F0  CD0271    1080 PARM:    CALL NUMX
70F3  2B        1090          DEC  HL
70F4  D8        1100          RET  C
70F5  7B        1110          LD   A,E
70F6  B2        1120          OR   D
70F7  C9        1130          RET
70F8            1140          DEFS 10
                1150 ;
7102  110000    1160 NUMX:    LD   DE,#0000
7105  010000    1170 A04:     LD   BC,#0000
7108  CD5B70    1180          CALL RST10
710B  C8        1190          RET  Z
710C  D0        1200          RET  NC
710D  E5        1210          PUSH HL
710E  D630      1220          SUB  #30
7110  4F        1230          LD   C,A
7111  62        1240          LD   H,D
7112  6B        1250          LD   L,E
7113  29        1260          ADD  HL,HL
7114  29        1270          ADD  HL,HL
7115  29        1280          ADD  HL,HL
7116  19        1290          ADD  HL,DE
7117  19        1300          ADD  HL,DE
7118  09        1310          ADD  HL,BC
7119  EB        1320          EX   DE,HL
711A  21F5FF    1330          LD   HL,#FFF5
711D  E7        1340          RST  #20
711E  E1        1350          POP  HL
711F  D8        1360          RET  C
7120  C30571    1370          JP   A04
7123            1380          DEFS 10
                1390 ;
712D  D5        1400 PXLIN:   PUSH DE
712E  C1        1410          POP  BC
712F  ED533471  1420          LD   (A05+1),DE
7133  2A0000    1430 A05:     LD   HL,(#0000)
7136  7C        1440          LD   A,H
7137  B5        1450          OR   L
7138  37        1460          SCF
7139  C8        1470          RET  Z
713A  13        1480          INC  DE
713B  13        1490          INC  DE
713C  ED534271  1500          LD   (A06+2),DE
7140  ED5B0000  1510 A06:     LD   DE,(#0000)
7144  2A167B    1520          LD   HL,(T03)
7147  E7        1530          RST  #20
7148  EB        1540          EX   DE,HL
7149  ED434F71  1550          LD   (A07+2),BC
714D  ED5B0000  1560 A07:     LD   DE,(#0000)
7151  C8        1570          RET  Z
7152  3F        1580          CCF
7153  C9        1590          RET
7154            1600          DEFS 20
```

```
                1610 *E
7168  CD5B70    1620 LISTAPG: CALL RST10
716B  285E      1630          JR   Z,B07
716D  3850      1640          JR   C,B05
716F  FE3C      1650          CP   #3C
7171  202D      1660          JR   NZ,B04
7173  CDF070    1670          CALL PARM
7176  3802      1680          JR   C,B01
7178  2005      1690          JR   NZ,B02
717A  ED5B007B  1700 B01:     LD   DE,(T01)
717E  13        1710          INC  DE
717F  ED53167B  1720 B02:     LD   (T03),DE
7183  CD5B70    1730          CALL RST10
7186  FE2C      1740          CP   #2C
7188  2041      1750          JR   NZ,B07
718A  CD5B70    1760          CALL RST10
718D  303C      1770          JR   NC,B07
718F  CDBC72    1780          CALL INDICE
7192  ED4B167B  1790          LD   BC,(T03)
7196  229B71    1800          LD   (B03+2),HL
7199  ED430000  1810 B03:     LD   (#0000),BC
719D  C3CC70    1820          JP   RET04
71A0  FE5B      1830 B04:     CP   #5B
71A2  2027      1840          JR   NZ,B07
71A4  110000    1850          LD   DE,#0000
71A7  ED53167B  1860          LD   (T03),DE
71AB  181E      1870          JR   B07
71AD            1880          DEFS 18
71BF  CDBC72    1890 B05:     CALL INDICE
71C2  22C671    1900          LD   (B06+1),HL
71C5  2A0000    1910 B06:     LD   HL,(#0000)
71C8  22167B    1920          LD   (T03),HL
71CB  3E0C      1930 B07:     LD   A,#0C
71CD  DF        1940          RST  #18
                1950 ;
71CE  ED5B76F6  1960 LISTPG1: LD   DE,(#F676)
71D2  CD2D71    1970 B10:     CALL PXLIN
71D5  302A      1980          JR   NC,LISTPG2
71D7  20F9      1990          JR   NZ,B10
                2000 ;
71D9  AF        2010 FIMLST:  XOR  A
71DA  32227B    2020          LD   (T07),A
71DD  211500    2030          LD   HL,#0015
71E0  C39D70    2040          JP   RET02
71E3            2050          DEFS 30
                2060 ;
7201  22007B    2070 LISTPG2: LD   (T01),HL
7204  CDB700    2080 B11:     CALL #00B7
7207  38D0      2090          JR   C,FIMLST
7209  D5        2100          PUSH DE
720A  CD2B72    2110          CALL IMPLIN
720D  D1        2120          POP  DE
720E  38C9      2130          JR   C,FIMLST
7210  CD2D71    2140          CALL PXLIN
7213  20EF      2150          JR   NZ,B11
7215  18C2      2160          JR   FIMLST
7217            2170          DEFS 20
```

```
                2180 *E
722B 22167B     2190 IMPLIN:  LD   (T03),HL
722E 210400     2200          LD   HL,#0004
7231 09         2210          ADD  HL,BC
7232 018452     2220          LD   BC,#5284
7235 CD8670     2230          CALL CALL
7238 3A227B     2240          LD   A,(T07)
723B FE00       2250          CP   #00
723D 2805       2260          JR   Z,B14
723F CD3B76     2270          CALL VERCHR
7242 3F         2280          CCF
7243 D0         2290          RET  NC
7244 2A167B     2300 B14:     LD   HL,(T03)
7247 011234     2310          LD   BC,#3412
724A CD8670     2320          CALL CALL
724D 3E20       2330          LD   A,#20
724F DF         2340          RST  #18
7250 215DF5     2350          LD   HL,#F55D
7253 CD6B72     2360          CALL IMPL
7256 C9         2370          RET
7257            2380          DEFS 20
                2390 ;
726B CD5B70     2400 IMPL:    CALL RST10
726E 2806       2410          JR   Z,B16
7270 CD9672     2420          CALL IMPTELA
7273 D8         2430          RET  C
7274 18F5       2440          JR   IMPL
7276 3E0A       2450 B16:     LD   A,#0A
7278 CD9672     2460          CALL IMPTELA
727B D8         2470          RET  C
727C 3E01       2480          LD   A,#01
727E 32DDF3     2490          LD   (#F3DD),A
7281 C9         2500          RET
7282            2510          DEFS 20
                2520 ;
7296 4F         2530 IMPTELA: LD   C,A
7297 3A467B     2540          LD   A,(T18)
729A FE00       2550          CP   #00
729C 2007       2560          JR   NZ,B17
729E 3ADCF3     2570          LD   A,(#F3DC)
72A1 FE17       2580          CP   #17
72A3 3F         2590          CCF
72A4 D8         2600          RET  C
72A5 79         2610 B17:     LD   A,C
72A6 DF         2620          RST  #18
72A7 C9         2630          RET
72A8            2640          DEFS 20
                2650 ;
72BC D630       2660 INDICE:  SUB  #30
72BE 87         2670          ADD  A,A
72BF 1600       2680          LD   D,#00
72C1 5F         2690          LD   E,A
72C2 21027B     2700          LD   HL,T02
72C5 19         2710          ADD  HL,DE
72C6 C9         2720          RET
72C7            2730          DEFS 10
```

```
                2740 *E
72D1 110300     2750 VOLTLIN: LD   DE,#0003
72D4 ED53187B   2760          LD   (T04),DE
72D8 CDF070     2770          CALL PARM
72DB 3810       2780          JR   C,C02
72DD 200A       2790          JR   NZ,C01
72DF CD5B70     2800          CALL RST10
72E2 FE5B       2810          CP   #5B
72E4 2007       2820          JR   NZ,C02
72E6 110100     2830          LD   DE,#0001
72E9 ED53187B   2840 C01:     LD   (T04),DE
72ED ED5B76F6   2850 C02:     LD   DE,(#F676)
72F1 2A007B     2860          LD   HL,(T01)
72F4 22167B     2870          LD   (T03),HL
72F7 3E0C       2880          LD   A,#0C
72F9 DF         2890          RST  #18
72FA CD2D71     2900 C03:     CALL PXLIN
72FD 3005       2910          JR   NC,C04
72FF 20F9       2920          JR   NZ,C03
7301 C3CC70     2930          JP   RET04
7304 C5         2940 C04:     PUSH BC
7305 D1         2950          POP  DE
7306 2A76F6     2960 C05:     LD   HL,(#F676)
7309 E7         2970          RST  #20
730A 201E       2980          JR   NZ,C06
730C 110000     2990          LD   DE,#0000
730F ED53167B   3000          LD   (T03),DE
7313 C3CE71     3010          JP   LISTPG1
7316            3020          DEFS 20
732A D5         3030 C06:     PUSH DE
732B 1B         3040 C07:     DEC  DE
732C 1B         3050 C08:     DEC  DE
732D 1A         3060          LD   A,(DE)
732E FE00       3070          CP   #00
7330 20FA       3080          JR   NZ,C08
7332 13         3090          INC  DE
7333 ED533873   3100          LD   (C09+1),DE
7337 2A0000     3110 C09:     LD   HL,(#0000)
733A C1         3120          POP  BC
733B A7         3130          AND  A
733C ED42       3140          SBC  HL,BC
733E C5         3150          PUSH BC
733F 20EA       3160          JR   NZ,C07
7341 C1         3170          POP  BC
7342 ED4B187B   3180          LD   BC,(T04)
7346 0B         3190          DEC  BC
7347 79         3200          LD   A,C
7348 B0         3210          OR   B
7349 281A       3220          JR   Z,C10
734B ED43187B   3230          LD   (T04),BC
734F 18B5       3240          JR   C05
7351            3250          DEFS 20
                3260 ;
```

```
7365 D5        3270 C10:     PUSH DE
7366 13        3280          INC  DE
7367 13        3290          INC  DE
7368 ED536D73  3300          LD   (C11+1),DE
736C 2A0000    3310 C11:     LD   HL,(#0000)
736F C1        3320          POP  BC
7370 ED437673  3330          LD   (C12+2),BC
7374 ED5B0000  3340 C12:     LD   DE,(#0000)
7378 C30172    3350          JP   LISTPG2

7500           3360          ORG  #7500
               3370 ;
7500 CD5B70    3380 NOVOBAS: CALL RST10
7503 FE72      3390          CP   #72
7505 CA6B75    3400          JP   Z,RETBAS
7508 FE75      3410          CP   #75
750A CAAC75    3420          JP   Z,UNIBAS
750D ED4B207B  3430 D05:     LD   BC,(T06)
7511 78        3440          LD   A,B
7512 B1        3450          OR   C
7513 C2B970    3460          JP   NZ,RET03
7516 ED4B76F6  3470          LD   BC,(#F676)
751A ED432075  3480          LD   (D06+2),BC
751E ED4B0000  3490 D06:     LD   BC,(#0000)
7522 78        3500          LD   A,B
7523 B1        3510          OR   C
7524 CAB970    3520          JP   Z,RET03
7527 ED5B76F6  3530          LD   DE,(#F676)
752B ED53207B  3540          LD   (T06),DE
752F CD2D71    3550 D07:     CALL PXLIN
7532 30FB      3560          JR   NC,D07
7534 20F9      3570          JR   NZ,D07
7536 ED4376F6  3580          LD   (#F676),BC
753A 0B        3590          DEC  BC
753B ED4348FC  3600          LD   (#FC48),BC
753F 211A7B    3610 D08:     LD   HL,T05
7542 115EF5    3620          LD   DE,#F55E
7545 010600    3630          LD   BC,#06
7548 EDB0      3640          LDIR
754A C3B970    3650          JP   RET03
754D           3660          DEFS 30
               3670 ;
756B ED4B207B  3680 RETBAS:  LD   BC,(T06)
756F 78        3690          LD   A,B
7570 B1        3700          OR   C
7571 CAB970    3710          JP   Z,RET03
7574 210000    3720          LD   HL,#0000
7577 22207B    3730          LD   (T06),HL
757A 2A76F6    3740          LD   HL,(#F676)
757D ED4376F6  3750          LD   (#F676),BC
7581 0B        3760          DEC  BC
7582 ED4348FC  3770          LD   (#FC48),BC
7586 3600      3780          LD   (HL),#00
7588 23        3790          INC  HL
7589 3600      3800          LD   (HL),#00
758B C33F75    3810          JP   D08
758E           3820          DEFS 30
               3830 ;
75AC ED4B207B  3840 UNIBAS:  LD   BC,(T06)
75B0 78        3850          LD   A,B
75B1 B1        3860          OR   C
75B2 CAB970    3870          JP   Z,RET03
75B5 210000    3880          LD   HL,#0000
75B8 22207B    3890          LD   (T06),HL
75BB ED4376F6  3900          LD   (#F676),BC
75BF 0B        3910          DEC  BC
75C0 ED4348FC  3920          LD   (#FC48),BC
75C4 C33F75    3930          JP   D08
75C7           3940          DEFS 30
```

```
                  3950 #E
75E5  CD5B70      3960 VERCTE:    CALL RST10
75E8  CAB970      3970            JP   Z,RET03
75EB  FE3D        3980            CP   #3D
75ED  281D        3990            JR   Z,E03
75EF  010000      4000            LD   BC,#0000
75F2  ED43167B    4010            LD   (T03),BC
75F6  11247B      4020            LD   DE,T09
75F9  12          4030            LD   (DE),A
75FA  13          4040            INC  DE
75FB  CD5B70      4050 E01:       CALL RST10
75FE  280A        4060            JR   Z,E02
7600  12          4070            LD   (DE),A
7601  13          4080            INC  DE
7602  1A          4090            LD   A,(DE)
7603  FEFF        4100            CP   #FF
7605  CAB970      4110            JP   Z,RET03
7608  18F1        4120            JR   E01
760A  AF          4130 E02:       XOR  A
760B  12          4140            LD   (DE),A
760C  3EFF        4150 E03:       LD   A,#FF
760E  32227B      4160            LD   (T07),A
7611  3E0C        4170            LD   A,#0C
7613  DF          4180            RST  #18
7614  21227B      4190            LD   HL,T08-1
7617  CD6B72      4200            CALL IMPL
761A  C3CE71      4210            JP   LISTP61
761D              4220            DEFS 30
                  4230 ;
763B  215DF5      4240 VERCHR:    LD   HL,#F55D
763E  11237B      4250 E10:       LD   DE,T08
7641  E5          4260            PUSH HL
7642  23          4270 E11:       INC  HL
7643  13          4280            INC  DE
7644  1A          4290            LD   A,(DE)
7645  FE00        4300            CP   #00
7647  281B        4310            JR   Z,E13
7649  4F          4320            LD   C,A
764A  7E          4330            LD   A,(HL)
764B  FE00        4340            CP   #00
764D  37          4350            SCF
764E  2814        4360            JR   Z,E13
7650  B9          4370            CP   C
7651  28EF        4380            JR   Z,E11
7653  FE41        4390            CP   #41
7655  3809        4400            JR   C,E12
7657  FE5B        4410            CP   #5B
7659  3005        4420            JR   NC,E12
765B  C620        4430            ADD  A,#20
765D  B9          4440            CP   C
765E  28E2        4450            JR   Z,E11
7660  E1          4460 E12:       POP  HL
7661  23          4470            INC  HL
7662  18DA        4480            JR   E10
7664  E1          4490 E13:       POP  HL
7665  C9          4500            RET
7666              4510            DEFS 30
                  4520 #E
7684  32427B      4530 COPMOV:    LD   (T15),A
7687  CDF070      4540            CALL PARM
768A  DAB970      4550            JP   C,RET03
768D  ED53397B    4560            LD   (T10),DE
7691  13          4570            INC  DE
7692  ED533B7B    4580            LD   (T11),DE
7696  CD5B70      4590            CALL RST10
7699  FE2C        4600            CP   #2C
769B  C2B970      4610            JP   NZ,RET03
769E  CDF070      4620            CALL PARM
76A1  DAB970      4630            JP   C,RET03
76A4  ED533D7B    4640            LD   (T12),DE
76A8  CD5B70      4650            CALL RST10
76AB  FE2C        4660            CP   #2C
76AD  2011        4670            JR   NZ,H00
76AF  D5          4680            PUSH DE
76B0  CDF070      4690            CALL PARM
76B3  C1          4700            POP  BC
76B4  DAC076      4710            JP   C,H00
76B7  03          4720            INC  BC
76B8  ED433B7B    4730            LD   (T11),BC
76BC  ED533D7B    4740            LD   (T12),DE
76C0  ED53167B    4750 H00:       LD   (T03),DE
76C4  ED5B76F6    4760            LD   DE,(#F676)
76C8  CD2D71      4770 H01:       CALL PXLIN
76CB  3005        4780            JR   NC,H02
76CD  CACC70      4790            JP   Z,RET04
76D0  18F6        4800            JR   H01
76D2  C2CC70      4810 H02:       JP   NZ,RET04
76D5  CD2D71      4820            CALL PXLIN
76D8  3023        4830            JR   NC,H03
76DA  01F5FF      4840            LD   BC,#FFF5
76DD  1828        4850            JR   H05
76DF              4860            DEFS 30
76FD  03          4870 H03:       INC  BC
76FE  03          4880            INC  BC
76FF  ED430577    4890            LD   (H04+2),BC
7703  ED4B0000    4900 H04:       LD   BC,(#0000)
7707  ED433F7B    4910 H05:       LD   (T13),BC
770B  ED4B397B    4920            LD   BC,(T10)
770F  0B          4930            DEC  BC
7710  ED43167B    4940            LD   (T03),BC
7714  AF          4950            XOR  A
7715  3214F4      4960            LD   (#F414),A
```

```
                4970 *E
7718  3EC9      4980 CML01:  LD   A,#C9
771A  320CFF    4990         LD   (#FF0C),A
771D  32477B    5000         LD   (T19),A
7720  CDB700    5010         CALL #00B7
7723  DACC70    5020         JP   C,RET04
7726  3A14F4    5030         LD   A,(#F414)
7729  FE00      5040         CP   #00
772B  C2CC70    5050         JP   NZ,RET04
772E  ED5B76F6  5060         LD   DE,(#F676)
7732  CD2D71    5070 H07:    CALL PXLIN
7735  3005      5080         JR   NC,H08
7737  CACC70    5090         JP   Z,RET04
773A  18F6      5100         JR   H07
773C  28F4      5110 H08:    JR   Z,H07
773E  22007B    5120         LD   (T01),HL
7741  ED5B3B7B  5130         LD   DE,(T11)
7745  E7        5140         RST  #20
7746  D2CC70    5150         JP   NC,RET04
7749  ED5B3F7B  5160         LD   DE,(T13)
774D  2A3D7B    5170         LD   HL,(T12)
7750  23        5180         INC  HL
7751  223D7B    5190         LD   (T12),HL
7754  E7        5200         RST  #20
7755  D2CC70    5210         JP   NC,RET04
7758  3A427B    5220         LD   A,(T15)
775B  32467B    5230         LD   (T18),A
775E  FE63      5240         CP   #63
7760  282B      5250         JR   Z,H11
7762  C5        5260         PUSH BC
7763  C5        5270         PUSH BC
7764  DDE1      5280         POP  IX
7766  01F654    5290         LD   BC,#54F6
7769  CD8670    5300         CALL CALL
776C  ED4B3D7B  5310         LD   BC,(T12)
7770  DD7102    5320         LD   (IX+2),C
7773  DD7003    5330         LD   (IX+3),B
7776  01F754    5340         LD   BC,#54F7
7779  CD8670    5350         CALL CALL
777C  ED4B007B  5360         LD   BC,(T01)
7780  DD7102    5370         LD   (IX+2),C
7783  DD7003    5380         LD   (IX+3),B
7786  C1        5390         POP  BC
7787  2A3D7B    5400         LD   HL,(T12)
778A  CD2B72    5410 H11:    CALL IMPLIN
778D  2A007B    5420         LD   HL,(T01)
7790  22167B    5430         LD   (T03),HL
7793  AF        5440         XOR  A
7794  32467B    5450         LD   (T18),A
7797  21DCF3    5460         LD   HL,#F3DC
779A  35        5470         DEC  (HL)
779B  3EC3      5480         LD   A,#C3
779D  320CFF    5490         LD   (#FF0C),A
77A0  32477B    5500         LD   (T19),A
77A3  C39A70    5510         JP   RET01
77A6            5520         DEFS 30

                5530 *E
77C4  3A427B    5540 CML02:  LD   A,(T15)
77C7  FE63      5550         CP   #63
77C9  CA1877    5560         JP   Z,CML01
77CC  3A417B    5570         LD   A,(T14)
77CF  EEFF      5580         XOR  #FF
77D1  32417B    5590         LD   (T14),A
77D4  CA1877    5600         JP   Z,CML01
77D7  3E20      5610         LD   A,#20
77D9  DF        5620         RST  #18
77DA  2A007B    5630         LD   HL,(T01)
77DD  011234    5640         LD   BC,#3412
77E0  CD8670    5650         CALL CALL
77E3  C39A70    5660         JP   RET01
77E6            5670         DEFS 30
```

```
               5680 *E
7804  CDEC78   5690 SINTAXE:  CALL ENDE
7807  01007F   5700 I01:      LD   BC,T21
780A  FEFA     5710           CP   #FA
780C  2004     5720           JR   NZ,I02
780E  2B       5730           DEC  HL
780F  110A60   5740           LD   DE,ERR
7812  1A       5750 I02:      LD   A,(DE)
7813  13       5760           INC  DE
7814  FE00     5770           CP   #00
7816  CAA178   5780           JP   Z,I09
7819  FEFF     5790           CP   #FF
781B  2845     5800           JR   Z,I05
781D  FEFE     5810           CP   #FE
781F  286D     5820           JR   Z,I07
7821  FEFC     5830           CP   #FC
7823  2876     5840           JR   Z,I08
7825  FEF8     5850           CP   #F8
7827  2025     5860           JR   NZ,I03
7829  7E       5870           LD   A,(HL)
782A  32457B   5880           LD   (T17),A
782D  23       5890           INC  HL
782E  1832     5900           JR   I05
7830           5910           DEFS 30
784E  FEF9     5920 I03:      CP   #F9
7850  2003     5930           JR   NZ,I04
7852  3A457B   5940           LD   A,(T17)
7855  FEFB     5950 I04:      CP   #FB
7857  2804     5960           JR   Z,I4A
7859  02       5970           LD   (BC),A
785A  03       5980           INC  BC
785B  18B5     5990           JR   I02
785D  324D7B   6000 I4A:      LD   (T20),A
7860  18B0     6010           JR   I02
7862  7E       6020 I05:      LD   A,(HL)
7863  FE00     6030 I06:      CP   #00
7865  28AB     6040           JR   Z,I02
7867  23       6050           INC  HL
7868  FE2C     6060           CP   #2C
786A  28A6     6070           JR   Z,I02
786C  02       6080           LD   (BC),A
786D  03       6090           INC  BC
786E  18F2     6100           JR   I05
7870           6110           DEFS 30
788E  7E       6120 I07:      LD   A,(HL)
788F  FE63     6130           CP   #63
7891  CC2B79   6140           CALL Z,PRIMLIN
7894  FE66     6150           CP   #66
7896  CC6379   6160           CALL Z,ULTLIN
7899  18C7     6170           JR   I05
               6180 *E
789B  7E       6190 I08:      LD   A,(HL)
789C  FE00     6200           CP   #00
789E  C21278   6210           JP   NZ,I02
78A1  02       6220 I09:      LD   (BC),A
78A2  ED4B4D7B 6230           LD   BC,(T20)
78A6  324D7B   6240           LD   (T20),A
78A9  B9       6250           CP   C
78AA  21007F   6260           LD   HL,T21
78AD  115EF5   6270           LD   DE,#F55E
78B0  01FA00   6280           LD   BC,#00FA
78B3  EDB0     6290           LDIR
78B5  12       6300           LD   (DE),A
78B6  CAB970   6310           JP   Z,RET03
78B9  215EF5   6320           LD   HL,#F55E
78BC  7E       6330 I10:      LD   A,(HL)
78BD  23       6340           INC  HL
78BE  FE00     6350           CP   #00
78C0  CA9A70   6360           JP   Z,RET01
78C3  DF       6370           RST  #18
78C4  18F6     6380           JR   I10
78C6           6390           DEFS 30
               6400 ;
78E4  110060   6410 LISTLIN:  LD   DE,TN
78E7  3E00     6420           LD   A,#00
78E9  C30778   6430           JP   I01
```

```
                6440 *E
78EC 47         6450 ENDE:    LD   B,A
78ED 23         6460          INC  HL
78EE 7E         6470          LD   A,(HL)
78EF 4F         6480          LD   C,A
78F0 110E60     6490          LD   DE,TABELA
78F3 1A         6500          LD   A,(DE)
78F4 13         6510          INC  DE
78F5 B8         6520 I20:     CP   B
78F6 280B       6530          JR   Z,I22
78F8 FEFA       6540 I21:     CP   #FA
78FA C8         6550          RET  Z
78FB FE00       6560          CP   #00
78FD 1A         6570          LD   A,(DE)
78FE 13         6580          INC  DE
78FF 28F4       6590          JR   Z,I20
7901 18F5       6600          JR   I21
7903 1A         6610 I22:     LD   A,(DE)
7904 FE20       6620          CP   #20
7906 13         6630          INC  DE
7907 C8         6640          RET  Z
7908 B9         6650          CP   C
7909 20ED       6660          JR   NZ,I21
790B 23         6670          INC  HL
790C C9         6680          RET
790D            6690          DEFS 30
                6700 ;
792B E5         6710 PRIMLIN: PUSH HL
792C D5         6720          PUSH DE
792D C5         6730          PUSH BC
792E ED5B76F6   6740          LD   DE,(#F676)
7932 CD2D71     6750          CALL PXLIN
                6760 ;
7935 22F8F7     6770 I30:     LD   (#F7F8),HL
7938 E1         6780          POP  HL
7939 010000     6790          LD   BC,#0000
793C CDDB36     6800          CALL #36DB
793F 44         6810          LD   B,H
7940 4D         6820          LD   C,L
7941 D1         6830          POP  DE
7942 E1         6840          POP  HL
7943 23         6850          INC  HL
7944 C9         6860          RET
7945            6870          DEFS 30
                6880 ;
7963 E5         6890 ULTLIN:  PUSH HL
7964 D5         6900          PUSH DE
7965 C5         6910          PUSH BC
7966 ED5B76F6   6920          LD   DE,(#F676)
796A 22437B     6930 I40:     LD   (T16),HL
796D CD2D71     6940          CALL PXLIN
7970 30F8       6950          JR   NC,I40
7972 20F6       6960          JR   NZ,I40
7974 2A437B     6970          LD   HL,(T16)
7977 18BC       6980          JR   I30
```

05 - ROTINA CARREGAMENTO BIT-BASIC

Quando é executado o comando

BLOAD"A:BIT.ASS",R ou BLOAD"CAS:BIT",R

o programa BIT-BASIC é colocado na memória do seu MSX nos endereços entre #A500 e #C0FF.
O "Endereço de Execução" é #C020 e o caráter "R" do comando acima indica que o Z-80 deve passar a executar as Instruções em "Linguagem de Máquina" contidas a partir deste endereço.
Estas Instruções executam os procedimentos de movimentar o BIT-BASIC da RAM (#A500 a #C0FF) para a RAM "paralela à ROM" (#6000 a #7FFF).
São movimentados o "Código Objeto" do BIT-BASIC, a "Rotina de Comunicação" do BIT-BASIC com o Interpretador BASIC e a "Tabela de Sintaxe" do BIT-BASIC.

Além disto, a "Rotina de Carregamento" também efetua a colocação dos "Endereços de Desvio" nos GANCHOS utilizados e "Liga" o GANCHO da rotina PINLIN do BIOS (#FDDB), colocando neste ponto uma Instrução JUMP (#C3) para a rotina DESVIO do BIT-BASIC (#FDD9).

ROTINA ASSEMBLER

```
0010          ORG  #C020
0020          IN   A,(#A8)
0030          LD   (#B58C),A
0040          CP   #F0
0050          LD   A,#FC
0060          JR   NZ,X01
0070          LD   (#C018),A
0080          JR   X02
0090 X01:     LD   A,#A8
0100          LD   (#C018),A
0110 X02:     OUT  A,#A8
0120          LD   HL,#B500
0130          LD   DE,#7000
0140          LD   BC,#0A00
0150          LDIR
0160          LD   HL,#C000
0170          LD   DE,#FFD9
0180          LD   BC,#0020
0190          LDIR
0200          LD   HL,#BFF0
0210          LD   DE,#6000
0220          LD   BC,#000F
0230          LDIR
0240          LD   HL,#A500
0250          LD   DE,#600E
0260          LD   BC,#0FF2
0270          LDIR
```

```
0280        LD   HL,#BF90
0290        LD   DE,#7B00
0300        LD   BC,004C
0310        LDIR
0320        LD   A,#C3
0330        LD   (#FDDB),A
0340        LD   HL,#FFD9
0350        LD   (#FDDC),HL
0360        LD   HL,#FFEB
0370        LD   (#FF0D),HL
0380        LD   HL,(#F676)
0390        LD   (#C089),HL
0400        LD   HL,#0000
0410        LD   (#8001),HL
0420        LD   A,(#B58C)
0430        OUT  (#A8),A
0440        RET
```

0010 Indica ao Compilador ASSEMBLER para colocar a rotina de carregamento a partir do endereço #C020.

0020 Coloca em REG-A o valor que determina a "Configuração" atual dos Bancos de Memória ROM/RAM (controlada pela PPI). Este valor é #F0 para o HOTBIT e #A0 para o EXPERT, em condições normais.

0030 Coloca REG-A no endereço #B58C que neste momento corresponde ao operando da Instrução 0790 da rotina JUMP do BIT-BASIC, utilizada para "Retornar" a configuração normal dos bancos de memória ROM/RAM, via Rotina DESVIO.

**** As Instruções 0040 a 0090 obtém o valor que deve substituir na PPI o valor "Original" (ROM BASIC-BIOS ATIVA) para que seja "Ativado" o banco de memória RAM "Paralela" à memória ROM nos endereços #6000 a #7FFF, onde será colocado o BIT-BASIC.
Estes valores são #FC (HOTBIT) ou #A8 (EXPERT).

0040 Compara REG-A (valor da Configuração Atual) com #F0.

0050 Coloca o valor #FC em REG-A.

0060 Se o valor anterior de REG-A (colocado em #B08C) não era #F0 (HOTBIT), desvia para a Instrução 0090 (X01).

0070 Coloca REG-A (#FC - HOTBIT) no endereço #C018, que corresponde ao operando da Instrução 0170 (DV02) da rotina DESVIO, destinada a "Ativar" a memória RAM do BIT-BASIC (#4000 a #7FFF).

0080 Desvia para a Instrução 0110 (X02).

0090 Coloca em REG-A o valor #A8, correspondente à Configuração desejada (EXPERT).

0100 Idem Instrução 0070, valor #A8 (EXPERT).

0110 "Ativa" a memória RAM "Paralela" à memória ROM, nos endereços #4000 a #7FFF. Para isto, REG-A contém neste momento o valor #FC (HOTBIT) ou #A8 (EXPERT).

0120 Carrega #B500 em REG-HL (Endereço DE).

0130 Carrega #7000 em REG-DE (Endereço PARA).

0140 Carrega #0A00 em REG-BC (Número de BYTES).

0150 Move 2560 BYTES (#0A00), a partir do endereço #B500, para o endereço #7000 (Programa BIT-BASIC).
0160 Carrega #C000 em REG-HL (Endereço DE).
0170 Carrega #FFD9 em REG-DE (Endereço PARA).
0180 Carrega #0020 em REG-BC (Número de BYTES).
0190 Move 32 BYTES (#0020), a partir do endereço #C000, para o endereço #FFD9 (Rotina DESVIO, de comunicação do BIT-BASIC com o BASIC).
0200 Carrega #BFF0 em REG-HL (Endereço DE).
0210 Carrega #6000 em REG-DE (Endereço PARA).
0220 Carrega #000F em REG-BC (Número de BYTES).
0230 Move 15 BYTES (#000F), a partir do endereço #BFF0, para o endereço #6000 (Primeira entrada da TABELA DE SINTAXE).
0240 Carrega #A500 em REG-HL (Endereço DE).
0250 Carrega #600E em REG-DE (Endereço PARA).
0260 Carrega #0FF2 em REG-BC (Número de BYTES).
0270 Move 4082 BYTES (#0FF2), a partir do endereço #A500, para o endereço #600E (Demais entradas na TABELA DE SINTAXE).
0280 Carrega #BF90 em REG-HL (Endereço DE).
0290 Carrega #7B00 em REG-DE (Endereço PARA).
0300 Carrega #004C em REG-BC (Número de BYTES).
0310 Move 76 BYTES (#004C), a partir do endereço #BF90, para o endereço #7B00 (Valores iniciais da Áreas de Trabalho).
0320 Coloca #C3 em REG-A (Código Instrução JUMP do Z-80).
0330 Coloca REG-A (#C3) no endereço #FDDB (Primeiro BYTE do GANCHO de PINLIN).
0340 Coloca #FFD9 em REG-HL.
0350 Coloca #FFD9 (que é o endereço da rotina DESVIO do BIT-BASIC) em #FDDC. Assim, teremos a Instrução Z-80 "#C3 #D9 #FF" (JUMP #FFD9) instalada no GANCHO da rotina PINLIN (Esta Instrução é aí colocada para provocar um desvio incondicional para a rotina DESVIO do BIT-BASIC, a cada linha entrada pelo usuário no modo DIRETO.)
0360 Coloca #FFEB em REG-HL.
0370 Coloca #FFEB no endereço #FF0D (#FF0D é o GANCHO da rotina MAIN-ENTRY do BASIC - #FFEB corresponde à Instrução 0140 (DV01) da rotina DESVIO).
Este GANCHO será "Ligado" em outro ponto do BIT-BASIC, quando extiver executando a rotina COPMOV.
0380 Coloca #F676 em REG-HL (endereço início programa BASIC).
0390 Coloca REG-HL no operando de endereço da instrução 0410, que fica então "LD (#8001),HL".
0400 Coloca #0000 em REG-HL.
0410 Carrega #0000 nas duas primeiras posições do Programa BASIC (o que indica a condição de "nenhum programa carregado". Por isto, após o carregamento do BIT-BASIC não haverá nenhum programa BASIC carregado).
0420 Recupera em REG-A a "Configuração Normal" dos Bancos de Memória ROM/RAM, aí colocada pela Instrução 0030.
0430 Retorna a Configuração de memória ao "Normal" (Interpretador BASIC ativo).
0440 RETORNA o controle ao Interpretador BASIC, com a rotina DESVIO interceptando e analisando antes dele os comandos por você teclados.

06 - CARACTERES DE CONTROLE BASIC

Se você estiver trabalhando com o BASIC é possível comandar algumas funções "Especiais" de "Edição de Tela" acionando a tecla CTRL (CONTROL = CONTROLE) simultaneamente a uma segunda tecla que indica qual a função a ser executada.
Você pode acionar estas funções a partir de um Programa ASSEMBLER, utilizando a técnica descrita no item 06 (Rotina RET01).
A seguir estão relacionadas as teclas e funções correspondentes, assim como os caracteres hexadecimais que as representam.

```
CTRL+A (#01) - Indica que o caráter seguinte é "Gráfico"
CTRL+B (#02) - Coloca o CURSOR no início da palavra anterior
CTRL+C (#03) - Encerra a condição de "Entrada de Dados"
CTRL+E (#05) - Apaga a linha desde o CURSOR até o final
CTRL+F (#06) - Coloca o CURSOR no início da palavra seguinte
CTRL+G (#07) - Aciona a rotina BEEP (Alarme) do BIOS
CTRL+H (#08) - Retrocede em uma posição os caracteres da linha,
               desde o CURSOR até o último caráter (equivale ao
               pressionamento da tecla "Back-Space")
CRTL+I (#09) - Coloca o CURSOR na posição seguinte de TABULAÇÃO
CTRL+J (#0A) - Coloca o CURSOR na linha seguinte (LINE FEED)
CTRL+K (#0B) - Coloca o CURSOR na posição (1,1) da Tela
CTRL+L (#0C) - Limpa a Tela e coloca o CURSOR na posição (1,1)
               (equivale ao acionamento da tecla "CLS/HOME")
CTRL+M (#0D) - Equivale ao pressionamento da tecla RETURN
               ("Retorno do Carro")
CTRL+N (#0E) - Coloca o CURSOR na última posição da linha
CTRL+R (#12) - Liga/Desliga o modo "Inserção" (equivale ao
               pressionamento da tecla INS)
CTRL+U (#15) - "Apaga" toda a linha sobre a qual está o CURSOR
CTRL+X (#18) - Equivale ao pressionamento da tecla SELECT
CTRL+[ (#1B) - Equivale ao pressionamento da tecla ESC
CTRL+\ (#1C) - Move o CURSOR uma posição para a Direita
CTRL+] (#1D) - Move o CURSOR uma posição para a Esquerda
CTRL+^ (#1E) - Move o CURSOR uma linha para Cima
CTRL+_ (#1F) - Move o CURSOR uma linha para Baixo
       (#7F) - Apaga a letra que está sobre o CURSOR (equivale ao
               pressionamento da tecla DEL)
```

07 - TABELA INSTRUÇÕES ASSEMBLER

8E	ADC	A,(HL)
DD8E00	ADC	A,(IX+0)
FD8E00	ADC	A,(IY+0)
8F	ADC	A,A
88	ADC	A,B
89	ADC	A,C
8A	ADC	A,D
CE00	ADC	A,0
8B	ADC	A,E
8C	ADC	A,H
8D	ADC	A,L
ED4A	ADC	HL,BC
ED5A	ADC	HL,DE
ED6A	ADC	HL,HL
ED7A	ADC	HL,SP
86	ADD	A,(HL)
DD8600	ADD	A,(IX+0)
FD8600	ADD	A,(IY+0)
87	ADD	A,A
80	ADD	A,B
81	ADD	A,C
82	ADD	A,D
C600	ADD	A,0
83	ADD	A,E
84	ADD	A,H
85	ADD	A,L
09	ADD	HL,BC
19	ADD	HL,DE
29	ADD	HL,HL
39	ADD	HL,SP
DD09	ADD	IX,BC
DD19	ADD	IX,DE
DD29	ADD	IX,IX
DD39	ADD	IX,SP
FD09	ADD	IY,BC
FD19	ADD	IY,DE
FD29	ADD	IY,IY
FD39	ADD	IY,SP
A6	AND	(HL)
DDA600	AND	(IX+0)
FDA600	AND	(IY+0)
A7	AND	A
A0	AND	B
A1	AND	C
A2	AND	D
E600	AND	0
A3	AND	E
A4	AND	H
A5	AND	L
CB46	BIT	0,(HL)
DDCB0046	BIT	0,(IX+0)
FDCB0046	BIT	0,(IY+0)
CB47	BIT	0,A
CB40	BIT	0,B
CB41	BIT	0,C
CB42	BIT	0,D
CB43	BIT	0,E
CB44	BIT	0,H
CB45	BIT	0,L
CB4E	BIT	1,(HL)
DDCB004E	BIT	1,(IX+0)
FDCB004E	BIT	1,(IY+0)
CB4F	BIT	1,A
CB48	BIT	1,B
CB49	BIT	1,C
CB4A	BIT	1,D
CB4B	BIT	1,E
CB4C	BIT	1,H
CB4D	BIT	1,L
CB56	BIT	2,(HL)
DDCB0056	BIT	2,(IX+0)
FDCB0056	BIT	2,(IY+0)
CB57	BIT	2,A
CB50	BIT	2,B
CB51	BIT	2,C
CB52	BIT	2,D
CB53	BIT	2,E
CB54	BIT	2,H
CB55	BIT	2,L
CB5E	BIT	3,(HL)
DDCB005E	BIT	3,(IX+0)
FDCB005E	BIT	3,(IY+0)
CB5F	BIT	3,A
CB58	BIT	3,B
CB59	BIT	3,C
CB5A	BIT	3,D
CB5B	BIT	3,E
CB5C	BIT	3,H
CB5D	BIT	3,L
CB66	BIT	4,(HL)
DDCB0066	BIT	4,(IX+0)
FDCB0066	BIT	4,(IY+0)
CB67	BIT	4,A
CB60	BIT	4,B
CB61	BIT	4,C
CB62	BIT	4,D
CB63	BIT	4,E
CB64	BIT	4,H
CB65	BIT	4,L
CB6E	BIT	5,(HL)
DDCB006E	BIT	5,(IX+0)
FDCB006E	BIT	5,(IY+0)
CB6F	BIT	5,A
CB68	BIT	5,B
CB69	BIT	5,C
CB6A	BIT	5,D
CB6B	BIT	5,E
CB6C	BIT	5,H
CB6D	BIT	5,L
CB76	BIT	6,(HL)
DDCB0076	BIT	6,(IX+0)
FDCB0076	BIT	6,(IY+0)
CB77	BIT	6,A
CB70	BIT	6,B
CB71	BIT	6,C
CB72	BIT	6,D
FB	EI	
E3	EX	(SP),HL
DDE3	EX	(SP),IX
FDE3	EX	(SP),IY
08	EX	AF,AF'
EB	EX	DE,HL
D9	EXX	
76	HALT	
ED46	IM	0
ED56	IM	1
ED5E	IM	2
ED78	IN	A,(C)
DB00	IN	A,(#00)
ED40	IN	B,(C)
ED48	IN	C,(C)
ED50	IN	D,(C)
ED58	IN	E,(C)
ED60	IN	H,(C)
ED68	IN	L,(C)
34	INC	(HL)
DD3400	INC	(IX+0)
FD3400	INC	(IY+0)
3C	INC	A
04	INC	B
03	INC	BC
0C	INC	C
14	INC	D
13	INC	DE
1C	INC	E
24	INC	H
23	INC	HL
DD23	INC	IX
FD23	INC	IY
2C	INC	L
33	INC	SP
EDAA	IND	
EDBA	INDR	
EDA2	INI	
EDB2	INIR	
E9	JP	(HL)
DDE9	JP	(IX)
FDE9	JP	(IY)
C30000	JP	#0000
DA0000	JP	C,#0000
FA0000	JP	M,#0000
D20000	JP	NC,#0000
C20000	JP	NZ,#0000
F20000	JP	P,#0000
EA0000	JP	PE,#0000
E20000	JP	PO,#0000
CA0000	JP	Z,#0000
38FE	JR	C,$+0
18FE	JR	$+0
30FE	JR	NC,$+0
20FE	JR	NZ,$+0
28FE	JR	Z,$+0
320000	LD	(#0000),A
ED430000	LD	(#0000),BC
CB73	BIT	6,E
CB74	BIT	6,H
CB75	BIT	6,L
CB7E	BIT	7,(HL)
DDCB007E	BIT	7,(IX+0)
FDCB007E	BIT	7,(IY+0)
CB7F	BIT	7,A
CB78	BIT	7,B
CB79	BIT	7,C
CB7A	BIT	7,D
CB7B	BIT	7,E
CB7C	BIT	7,H
CB7D	BIT	7,L
CD0000	CALL	#0000
DC0000	CALL	C,#0000
FC0000	CALL	M,#0000
D40000	CALL	NC,#0000
C40000	CALL	NZ,#0000
F40000	CALL	P,#0000
EC0000	CALL	PE,#0000
E40000	CALL	PO,#0000
CC0000	CALL	Z,#0000
3F	CCF	
BE	CP	(HL)
DDBE00	CP	(IX+0)
FDBE00	CP	(IY+0)
BF	CP	A
B8	CP	B
B9	CP	C
BA	CP	D
FE00	CP	0
BB	CP	E
BC	CP	H
BD	CP	L
EDA9	CPD	
EDB9	CPDR	
EDA1	CPI	
EDB1	CPIR	
2F	CPL	
27	DAA	
35	DEC	(HL)
DD3500	DEC	(IX+0)
FD3500	DEC	(IY+0)
3D	DEC	A
05	DEC	B
0B	DEC	BC
0D	DEC	C
15	DEC	D
1B	DEC	DE
1D	DEC	E
25	DEC	H
2B	DEC	HL
DD2B	DEC	IX
FD2B	DEC	IY
2D	DEC	L
3B	DEC	SP
F3	DI	
10FE	DJNZ	$+0

Código	Instrução	Operandos
ED530000	LD	(#000),DE
220000	LD	(#0000),HL
DD220000	LD	(#0000),IX
FD220000	LD	(#0000),IY
ED730000	LD	(#0000),SP
02	LD	(BC),A
12	LD	(DE),A
77	LD	(HL),A
70	LD	(HL),B
71	LD	(HL),C
72	LD	(HL),D
3600	LD	(HL),0
73	LD	(HL),E
74	LD	(HL),H
75	LD	(HL),L
DD7700	LD	(IX+0),A
DD7000	LD	(IX+0),B
DD7100	LD	(IX+0),C
DD7200	LD	(IX+0),D
DD360000	LD	(IX+0),0
DD7300	LD	(IX+0),E
DD7400	LD	(IX+0),H
DD7500	LD	(IX+0),L
FD7700	LD	(IY+0),A
FD7000	LD	(IY+0),B
FD7100	LD	(IY+0),C
FD7200	LD	(IY+0),D
FD360000	LD	(IY+0),0
FD7300	LD	(IY+0),E
FD7400	LD	(IY+0),H
FD7500	LD	(IY+0),L
3A0000	LD	A,(#0000)
0A	LD	A,(BC)
1A	LD	A,(DE)
7E	LD	A,(HL)
DD7E00	LD	A,(IX+0)
FD7E00	LD	A,(IY+0)
7F	LD	A,A
78	LD	A,B
79	LD	A,C
7A	LD	A,D
3E00	LD	A,0
7B	LD	A,E
7C	LD	A,H
ED57	LD	A,I
7D	LD	A,L
ED5F	LD	A,R
46	LD	B,(HL)
DD4600	LD	B,(IX+0)
FD4600	LD	B,(IY+0)
47	LD	B,A
40	LD	B,B
41	LD	B,C
42	LD	B,D
0600	LD	B,0
43	LD	B,E
44	LD	B,H
45	LD	B,L
ED4B0000	LD	BC,(#0000)
010000	LD	BC,#00
4E	LD	C,(HL)
DD4E00	LD	C,(IX+0)
FD4E00	LD	C,(IY+0)
4F	LD	C,A
48	LD	C,B
49	LD	C,C
4A	LD	C,D
0E00	LD	C,0
4B	LD	C,E
4C	LD	C,H
4D	LD	C,L
56	LD	D,(HL)
DD5600	LD	D,(IX+0)
FD5600	LD	D,(IY+0)
57	LD	D,A
50	LD	D,B
51	LD	D,C
52	LD	D,D
1600	LD	D,0
53	LD	D,E
54	LD	D,H
55	LD	D,L
ED5B0000	LD	DE,(#0000)
110000	LD	DE,#0000
5E	LD	E,(HL)
DD5E00	LD	E,(IX+0)
FD5E00	LD	E,(IY+0)
5F	LD	E,A
58	LD	E,B
59	LD	E,C
5A	LD	E,D
1E00	LD	E,0
5B	LD	E,E
5C	LD	E,H
5D	LD	E,L
66	LD	H,(HL)
DD6600	LD	H,(IX+0)
FD6600	LD	H,(IY+0)
67	LD	H,A
60	LD	H,B
61	LD	H,C
62	LD	H,D
2600	LD	H,0
63	LD	H,E
64	LD	H,H
65	LD	H,L
2A0000	LD	HL,(#0000)
210000	LD	HL,#0000
ED47	LD	I,A
DD2A0000	LD	IX,(#0000)
DD210000	LD	IX,#0000
FD2A0000	LD	IY,(#0000)
FD210000	LD	IY,#0000
6F	LD	L,A
68	LD	L,B
69	LD	L,C
6A	LD	L,D
2E00	LD	L,0
6B	LD	L,E
6E	LD	L,(HL)
DD6E00	LD	L,(IX+0)
FD6E00	LD	L,(IY+0)
6C	LD	L,H
6D	LD	L,L
ED4F	LD	R,A
ED7B0000	LD	SP,(#0000)
310000	LD	SP,#00
F9	LD	SP,HL
DDF9	LD	SP,IX
FDF9	LD	SP,IY
EDA8	LDD	
EDB8	LDDR	
EDA0	LDI	
EDB0	LDIR	
ED44	NEG	
00	NOP	
B6	OR	(HL)
DDB600	OR	(IX+0)
FDB600	OR	(IY+0)
B7	OR	A
B0	OR	B
B1	OR	C
B2	OR	D
F600	OR	0
B3	OR	E
B4	OR	H
B5	OR	L
EDBB	OTDR	
EDB3	OTIR	
ED79	OUT	(C),A
ED41	OUT	(C),B
ED49	OUT	(C),C
ED51	OUT	(C),D
ED59	OUT	(C),E
ED61	OUT	(C),H
ED69	OUT	(C),L
D300	OUT	(#00),A
EDAB	OUTD	
EDA3	OUTI	
F1	POP	AF
C1	POP	BC
D1	POP	DE
E1	POP	HL
DDE1	POP	IX
FDE1	POP	IY
F5	PUSH	AF
C5	PUSH	BC
D5	PUSH	DE
E5	PUSH	HL
DDE5	PUSH	IX
FDE5	PUSH	IY
CB86	RES	0,(HL)
DDCB0086	RES	0,(IX+0)
FDCB0086	RES	0,(IY+0)
CB87	RES	0,A
CB80	RES	0,B
CB81	RES	0,C
CB82	RES	0,D
CB83	RES	0,E
CB84	RES	0,H
CB85	RES	0,L
CB8E	RES	1,(HL)
DDCB008E	RES	1,(IX+0)
FDCB008E	RES	1,(IY+0)
CB8F	RES	1,A
CB88	RES	1,B
CB89	RES	1,C
CB8A	RES	1,D
CB8B	RES	1,E
CB8C	RES	1,H
CB8D	RES	1,L
CB96	RES	2,(HL)
DDCB0096	RES	2,(IX+0)
FDCB0096	RES	2,(IY+0)
CB97	RES	2,A
CB90	RES	2,B
CB91	RES	2,C
CB92	RES	2,D
CB93	RES	2,E
CB94	RES	2,H
CB95	RES	2,L
CB9E	RES	3,(HL)
DDCB009E	RES	3,(IX+0)
FDCB009E	RES	3,(IY+0)
CB9F	RES	3,A
CB98	RES	3,B
CB99	RES	3,C
CB9A	RES	3,D
CB9B	RES	3,E
CB9C	RES	3,H
CB9D	RES	3,L
CBA6	RES	4,(HL)
DDCB00A6	RES	4,(IX+0)
FDCB00A6	RES	4,(IY+0)
CBA7	RES	4,A
CBA0	RES	4,B
CBA1	RES	4,C
CBA2	RES	4,D
CBA3	RES	4,E
CBA4	RES	4,H
CBA5	RES	4,L
CBAE	RES	5,(HL)
DDCB00AE	RES	5,(IX+0)
FDCB00AE	RES	5,(IY+0)
CBAF	RES	5,A
CBA8	RES	5,B
CBA9	RES	5,C
CBAA	RES	5,D
CBAB	RES	5,E
CBAC	RES	5,H
CBAD	RES	5,L
CBB6	RES	6,(HL)

DDCB0086	RES	6,(IX+0)
FDCB0086	RES	6,(IY+0)
CBB7	RES	6,A
CBB0	RES	6,B
CBB1	RES	6,C
CBB2	RES	6,D
CBB3	RES	6,E
CBB4	RES	6,H
CBB5	RES	6,L
CBBE	RES	7,(HL)
DDCB00BE	RES	7,(IX+0)
FDCB00BE	RES	7,(IY+0)
CBBF	RES	7,A
CBB8	RES	7,B
CBB9	RES	7,C
CBBA	RES	7,D
CBBB	RES	7,E
CBBC	RES	7,H
CBBD	RES	7,L
C9	RET	
D8	RET	C
F8	RET	M
D0	RET	NC
C0	RET	NZ
F0	RET	P
E8	RET	PE
E0	RET	PO
C8	RET	Z
ED4D	RETI	
ED45	RETN	
CB16	RL	(HL)
DDCB0016	RL	(IX+0)
FDCB0016	RL	(IY+0)
CB17	RL	A
CB10	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E
CB14	RL	H
CB15	RL	L
CB17	RL	A
CB06	RLC	(HL)
DDCB0006	RLC	(IX+0)
FDCB0006	RLC	(IY+0)
CB07	RLC	A
CB00	RLC	B
CB01	RLC	C
CB02	RLC	D
CB03	RLC	E
CB04	RLC	H
CB05	RLC	L
07	RLCA	
ED6F	RLD	
CB1E	RR	(HL)
DDCB001E	RR	(IX+0)
FDCB001E	RR	(IY+0)
CB1F	RR	A
CB18	RR	B

CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
1F	RRA	
CB0E	RRC	(HL)
DDCB000E	RRC	(IX+0)
FDCB000E	RRC	(IY+0)
CB0F	RRC	A
CB08	RRC	B
CB09	RRC	C
CB0A	RRC	D
CB0B	RRC	E
CB0C	RRC	H
CB0D	RRC	L
0F	RRCA	
ED67	RRD	
C7	RST	#00
CF	RST	#08
D7	RST	#10
DF	RST	#18
E7	RST	#20
EF	RST	#28
F7	RST	#30
FF	RST	#38
9E	SBC	A,(HL)
DD9E00	SBC	A,(IX+0)
FD9E00	SBC	A,(IY+0)
9F	SBC	A,A
98	SBC	A,B
99	SBC	A,C
9A	SBC	A,D
DE00	SBC	A,0
9B	SBC	A,E
9C	SBC	A,H
9D	SBC	A,L
ED42	SBC	HL,BC
ED52	SBC	HL,DE
ED62	SBC	HL,HL
ED72	SBC	HL,SP
37	SCF	
CBC6	SET	0,(HL)
DDCB00C6	SET	0,(IX+0)
FDCB00C6	SET	0,(IY+0)
CBC7	SET	0,A
CBC0	SET	0,B
CBC1	SET	0,C
CBC2	SET	0,D
CBC3	SET	0,E
CBC4	SET	0,H
CBC5	SET	0,L
CBCE	SET	1,(HL)
DDCB00CE	SET	1,(IX+0)
FDCB00CE	SET	1,(IY+0)
CBCF	SET	1,A
CBC8	SET	1,B
CBC9	SET	1,C

CBCA	SET	1,D
CBCB	SET	1,E
CBCC	SET	1,H
CBCD	SET	1,L
CBD6	SET	2,(HL)
DDCB00D6	SET	2,(IX+0)
FDCB00D6	SET	2,(IY+0)
CBD7	SET	2,A
CBD0	SET	2,B
CBD1	SET	2,C
CBD2	SET	2,D
CBD3	SET	2,E
CBD4	SET	2,H
CBD5	SET	2,L
CBDE	SET	3,(HL)
DDCB00DE	SET	3,(IX+0)
FDCB00DE	SET	3,(IY+0)
CBDF	SET	3,A
CBD8	SET	3,B
CBD9	SET	3,C
CBDA	SET	3,D
CBDB	SET	3,E
CBDC	SET	3,H
CBDD	SET	3,L
CBE6	SET	4,(HL)
DDCB00E6	SET	4,(IX+0)
FDCB00E6	SET	4,(IY+0)
CBE7	SET	4,A
CBE0	SET	4,B
CBE1	SET	4,C
CBE2	SET	4,D
CBE3	SET	4,E
CBE4	SET	4,H
CBE5	SET	4,L
CBEE	SET	5,(HL)
DDCB00EE	SET	5,(IX+0)
FDCB00EE	SET	5,(IY+0)
CBEF	SET	5,A
CBE8	SET	5,B
CBE9	SET	5,C
CBEA	SET	5,D
CBEB	SET	5,E
CBEC	SET	5,H
CBED	SET	5,L
CBF6	SET	6,(HL)
DDCB00F6	SET	6,(IX+0)
FDCB00F6	SET	6,(IY+0)
CBF7	SET	6,A
CBF0	SET	6,B
CBF1	SET	6,C
CBF2	SET	6,D
CBF3	SET	6,E
CBF4	SET	6,H
CBF5	SET	6,L
CBFE	SET	7,(HL)
DDCB00FE	SET	7,(IX+0)
FDCB00FF	SET	7,(IY+0)
CBFF	SET	7,A

CBF8	SET	7,B
CBF9	SET	7,C
CBFA	SET	7,D
CBFB	SET	7,E
CBFC	SET	7,H
CBFD	SET	7,L
CB26	SLA	(HL)
DDCB0026	SLA	(IX+0)
FDCB0026	SLA	(IY+0)
CB27	SLA	A
CB20	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB2E	SRA	(HL)
DDCB002E	SRA	(IX+0)
FDCB002E	SRA	(IY+0)
CB2F	SRA	A
CB28	SRA	B
CB29	SRA	C
CB2A	SRA	D
CB2B	SRA	E
CB2C	SRA	H
CB2D	SRA	L
CB3E	SRL	(HL)
DDCB003E	SRL	(IX+0)
FDCB003E	SRL	(IY+0)
CB3F	SRL	A
CB38	SRL	B
CB39	SRL	C
CB3A	SRL	D
CB3B	SRL	E
CB3C	SRL	H
CB3D	SRL	L
96	SUB	(HL)
DD9600	SUB	(IX+0)
FD9600	SUB	(IY+0)
97	SUB	A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
D600	SUB	0
94	SUB	H
95	SUB	L
AE	XOR	(HL)
DDAE00	XOR	(IX+0)
FDAE00	XOR	(IY+0)
AF	XOR	A
A8	XOR	B
A9	XOR	C
AA	XOR	D
EE00	XOR	0
AB	XOR	E
AC	XOR	H
AD	XOR	L

LANÇA ESTA GRANDE NOVIDADE PARA O SEU **MSX**:
BIT-BASIC
(Livro + Software)

Aqui está o livro que explica *detalhadamente*, instrução por instrução, o Software Bit-Basic com que você faz alguns "milagres"...

Novos e interessantíssimos comandos! • listagem controlada de programas • cópia/movimentação de linhas • busca de constantes • dois programas carregados (com possibilidades de "união" sem "misturar" as linhas).

Basic com *"Sintaxe Simplificada"*!

Programas Basic "de uma linha" previamente carregados! • programas utilitários, acionados com sintaxe simplificada (sem necessidade de carga a partir de fita/disquete), inclusive programas criados por você mesmo!

E tudo isto com o Interpretador Basic integralmente e simultaneamente disponível!

E mais:

Você também vai aprender essa mágica! O livro ensina como interceptar o Basic e alguns de seus misteriosos segredos. O Bit-Basic é uma raridade, pois é um Software "aberto" ao usuário e *completamente documentado*!

Você não conhece ASSEMBLER? Não tem problema! Este livro, que acompanha o Software Bit-Basic, explica o funcionamento do Z-80 e de sua "linguagem" (o Assembler Z-80) de maneira simples e objetiva, utilizando, para isso, os seus próprios conhecimentos do Basic!

BIT-BASIC
UM SOFTWARE
INTEIRAMENTE NACIONAL

BIT-BASIC

Luiz C. Bittencourt